Руководство по эксплуатации R-Vision VM

v3.0

Оглавление

О системе . . . 10
Развертывание системы . . . 13
- Технические требования . . . 13
- Развертывание кластера Kubernetes . . . 18
- Установка СУБД . . . 22
  - Установка ClickHouse на выделенном сервере . . . 22
    - Настройка нового кластера ClickHouse . . . 23
    - Настройка существующего кластера ClickHouse . . . 27
    - Справка о параметрах ClickHouse . . . 39
  - Установка PostgreSQL на выделенном сервере . . . 41
- Подготовка к установке системы . . . 43
- Установка системы . . . 45
  - Пакет установки . . . 46
  - Процесс установки системы . . . 52
- Установка сателлитов . . . 68
  - Технические требования к сателлитам . . . 69
  - Подготовка к установке сателлита . . . 71
  - Процесс установки сателлита . . . 73
- Обновление системы . . . 78
  - Обновление ClickHouse . . . 80
- Устранение проблем при установке или обновлении . . . 83
- Установка и удаление агентов . . . 84
  - Требования к установке агента . . . 84
  - Установка агента . . . 85
  - Настройка конфигурации агента . . . 88
  - Работа с установленными агентами . . . 92
  - Подключение агента к кластеру . . . 97
  - Подключение агента с шифрованием . . . 100
    - Добавление TLS-сертификата на агенте . . . 100
    - Настройка шифрования между агентом и шлюзом . . . 105
  - Обновление агента . . . 106
  - Устранение проблем с агентами . . . 108
  - Удаление агента . . . 109
- Удаление системы . . . 112

Администрирование системы . . . 113
- Добавление SSL-сертификата . . . 114
  - Добавление самоподписанного сертификата в список доверенных . . . 120
- Работа с кластером . . . 122
  - Подключение к кластеру . . . 123

Мониторинг кластера . . . 125
Состояние кластера . . . 125
Создание пользовательских дашбордов в Grafana . . . 128
Анализ работоспособности сервисов кластера . . . 131
Анализ работоспособности контрольной плоскости кластера . . . 132
Анализ состояния сателлита . . . 135
Просмотр журнала событий кластера . . . 136
Просмотр состояния подов кластера . . . 137
Список подов сервисов . . . 141
Настройка оповещений . . . 144
Настройка оповещений в Telegram . . . 148
Настройка оповещений по электронной почте . . . 150
Работа с метриками Prometheus . . . 152
Управление кластером . . . 158
Управление узлами кластера . . . 158
Работа с подом collector-manager . . . 162
Увеличение размера тома для подов . . . 164
Резервное копирование и восстановление PostgreSQL . . . 167
Резервное копирование хранилища etcd . . . 171
Получение идентификатора сервиса в NATS Service Discovery . . . 173
Работа с объектным хранилищем NATS Object Store . . . 174
Работа с паролями в кластере . . . 176
Изменение доменного имени установленного экземпляра системы . . . 179
Настройка системы . . . 182
Настройка интеграции с продуктами R-Vision . . . 182
Настройка формата даты . . . 183
Изменение пула адресов в MetalLB . . . 186
Настройка времени хранения запросов . . . 187
Управление лицензией . . . 188
Управление пространствами . . . 190
Работа с пространствами . . . 192
Работа с сервисами . . . 196
Управление доступом . . . 202
Тенанты . . . 202
Пользователи . . . 206
Группы пользователей . . . 215
Роли . . . 224
Стандартные роли . . . 229
Политики входа и пароля . . . 235
Настройка времени жизни для токенов доступа . . . 238
LDAP-соединения . . . 239
Основные пользовательские атрибуты Active Directory . . . 244

Политики управления доступом к событиям 245
Настройка агентов 249
Настройки менеджера агентов 250
Сбор инвентарной информации 250
Сбор событий на агентах 251
Аудит 252
Аудит источников событий 253
Политики аудита источников событий 253
Настройка конвейера для аудита коллектора 262
Аудит системных событий 270
Модель события аудита 279
Отслеживание действий суперадминистраторов 281
Отслеживание действий над объектами 284
Мониторинг RQL-запросов и управление ресурсами 287
Устранение проблем при эксплуатации системы 293
Авторизация 309
Интерфейс 310
Документация 313
Настройки профиля 313
Настройки интерфейса 314
Смена пароля 315
Работа с таблицами 316
Клавиши и комбинации клавиш 320
Начало работы с системой 322
Обработка событий на одном конвейере 322
Обработка событий на нескольких конвейерах 352
Обработка событий на нескольких коллекторах 359
Обработка событий с нескольких IP-адресов 365
Сбор событий из вынесенного коллектора 369
Автоматизация 377
Работа с запусками 377
Работа с триггерами 383
Работа с резолюциями 391
Аналитика 395
Работа с дашбордами 395
Работа с отчетами 417
Работа с разделом Поиск 427
Ресурсы 450
Работа с коллекторами 451
Работа с конвейерами 463
Работа с точками входа 475
Работа с VRL-трансформациями 499

Работа с фильтрами . . . 502
Работа с маршрутизаторами . . . 505
Работа с коррелляторами . . . 508
Работа с нормализаторами . . . 512
Работа с агрегаторами . . . 516
Работа со сбором метрик . . . 520
Работа с конечными точками . . . 522
Работа с шинами . . . 532
Работа с обогащениями . . . 536
Работа с моделями событий . . . 539
Универсальная модель события . . . 546
Типы данных . . . 559
Работа с базами данных событий . . . 564
Работа с хранилищами событий . . . 570
Работа с активными списками . . . 578
Работа с секретами . . . 584
Работа с доменами . . . 589
Редактор связей . . . 601
Работа с переменными . . . 608
Работа с конфигурациями . . . 612
Работа с воркерами . . . 617
Работа со сканерами . . . 628
Агенты . . . 630
Работа с агентами . . . 630
Работа с группами агентов . . . 640
Настройка конфигурации группы агентов . . . 645
Синтаксис языка XPath . . . 652
Работа со шлюзами . . . 655
Сканирование на уязвимости . . . 661
База уязвимостей R-Vision . . . 661
Интеграции с внешними системами . . . 661
Учетные записи . . . 683
Настройка прав учетной записи Windows . . . 683
Конфигурирование службы WinRM вручную . . . 684
Настройка службы WinRM с помощью групповых политик . . . 687
Настройка прав учетной записи Linux . . . 690
Работа с учетными записями . . . 692
Профили сканирования . . . 697
Вкладки профилей . . . 700
Задачи сканирования . . . 705
Безагентское сканирование . . . 706
Агентское сканирование . . . 711

Настройка отображения событий аудита, связанных со сканированием 714
Проверки соответствия 715
Проверки 715
База проверок 719
Управление уязвимостями 722
Уязвимости 723
Задачи по устранению уязвимостей 731
Изменение настроек интеграции с Jira 736
База уязвимостей 737
Менеджер обновлений 742
Обновление базы уязвимостей 742
Обновление файла проверки соответствия 744
Использование коллекции Postman 745
Экспертиза 746
Работа с элементами экспертизы 746
Правила корреляции 763
Построение правила корреляции в конструкторе 767
Императивные правила 781
Декларативные правила 807
Правила нормализации 850
Правила агрегации 864
Схемы активных списков 885
Схемы дашбордов 898
Схемы шаблонов отчетов 916
Таблицы обогащения 933
Шаблоны поиска 946
Глобальные функции 957
Схемы доменов 972
Структура схемы домена 975
Описания интерфейса 1023
Типы данных в схемах доменов 1081
Скрипты 1089
API в скриптах 1099
Плейбуки 1145
Модули плейбуков 1161
Специальные переменные в плейбуках 1172
Переменная results 1181
Схемы конфигурации 1186
Схемы расчетов 1191
Наследование в схемах расчетов 1203
Тестирование правил экспертизы с помощью языка VRL 1206
Справочник по языку запросов RQL 1212

Работа с RQL-запросами 1212
Функции 1234
Агрегатные функции 1235
Условные функции 1240
Функции сравнения 1240
Логические функции 1241
Функции преобразования типов 1242
Функции для работы с Nullable-аргументами 1253
Математические функции 1254
Арифметические функции 1256
Битовые функции 1258
Функции округления 1259
Функции для работы с датой и временем 1260
Функции для работы со строками 1262
Функции разбиения и слияния строк и массивов 1265
Функции для поиска в строках 1266
Функции для замены строк 1269
Функции для работы с массивами 1269
Географические функции 1273
Функции временных рядов 1276
Функции для работы с IP-адресами 1277
Функции для работы с URL 1279
Функции для работы с UUID 1280
Справочник по языку преобразования данных VRL 1282
О языке VRL 1282
Выражения 1287
Функции 1330
Функции разбора 1331
Функции кодирования и декодирования 1412
Функции событий и активных списков 1435
Функции коллекций 1445
Функции строк 1480
Функции типов и форматирования 1512
Хэш-функции и криптографические функции 1554
Функции генерации случайных значений и ID 1568
Системные функции 1576
Функции отладки 1578
Справочник ошибок 1582
Советы по написанию кода на языке VRL 1599
Оптимизация кода 1599
Оформление кода 1607
GraphQL API 1609

Открытие доступа к GraphQL-песочнице . . . 1612
API-токены . . . 1613
Работа с GraphQL . . . 1615
Запросы API модуля управления уязвимостями . . . 1638
Обработка ошибок API . . . 1732
Глоссарий . . . 1736

Документация по продукту R-Vision VM (также "система") содержит описание функциональных характеристик системы и сведения, необходимые для ее эксплуатации, включая пошаговые инструкции и рекомендации по настройке.

О системе

Система представляет собой модульный программный комплекс, предназначенный для моделирования и автоматизации бизнес-процессов с помощью модулей, решающих задачи в области информационных технологий и информационной безопасности (ИТ и ИБ).

Основные функциональные возможности системы

Управление модулями в распределенной инфраструктуре

- Размещение модулей комплекса в распределенной инфраструктуре.
- Управление пространствами и размещенными в них сервисами.

Управление доступом к функциям системы

- Поддержка мультитенантности — возможности создать в рамках одного экземпляра системы несколько логических копий системы (тенантов), каждый из которых конфигурируется независимо.
- Гибкое управление пользователями: настройка прав доступа на основе ролевой модели, задание политик входа и пароля.

Моделирование предметных областей

- Моделирование предметных областей (доменов), включая описание сущностей, их структур и взаимосвязей.
- Создание доменных представлений, позволяющих работать с доменом в целом и отображать списки сущностей.
- Создание представлений сущностей, определяющих структуру и вид пользовательского интерфейса для работы с экземпляром сущности.

Автоматизация процессов в системе

- Создание или импорт в систему скриптов — сценариев на поддерживаемых языках, которые определяют последовательность выполнения различных действий в системе.
- Создание или импорт в систему плейбуков — сложных взаимосвязанных наборов скриптов.
- Создание конфигураций, позволяющих описывать настройки соединения с внешними системами.
- Создание переменных и секретов для использования в скриптах и плейбуках.
- Создание триггеров для автоматического запуска скриптов и плейбуков при наступлении определенных событий или по расписанию.
- Создание воркеров, управляющих распределением ресурсов системы в зависимости от выполняемой задачи.
- Управление запусками — компонентами системы, позволяющими запускать скрипты и плейбуки, а также проверять статус их выполнения.

Обработка событий ИБ

- Сбор, обработка и хранение событий, полученных от поддерживаемых источников.
- Глубокий аналитический разбор событий, включающий их корреляцию, нормализацию и агрегацию.
- Создание хранилищ событий и пользовательских моделей событий.
- Выполнение аналитических запросов к событиям в хранилищах.
- Тестирование работоспособности объектов системы.
- Мониторинг и анализ системных событий и источников.
- Формирование виджетов и аналитических дашбордов для отслеживания показателей по событиям, а также метрик работы компонентов системы.
- Построение отчетов на основе аналитических данных.

Управление уязвимостями

- Сканирование оборудования в безагентском и агентском режимах.
- Настройка учетных записей (SSH, Windows) для аутентификации на оборудовании при сканировании оборудования и поиске уязвимостей.
- Настройка учетных записей для аудита СУБД различных типов (MS SQL, Oracle, PostgreSQL, ClickHouse).
- Создание и настройка сканеров для сбора сведений об оборудовании во внешних системах.
- Создание и настройка различных профилей сканирования, содержащих данные о технических параметрах сканирования узлов и сетей.
- Создание журнала сканирования для оценки успешности сканирования оборудования в нескольких режимах.
- Работа со сведениями об уязвимостях, обнаруженных системой при сканировании оборудования.
- Работа со сведениями, полученными при запуске задач сканирования по проверке стандартов.
- Ручное и автоматическое обновление базы уязвимостей, позволяющей проводить анализ уязвимостей по различным критериям.

Управление агентами на конечных устройствах

- Установка агентов на конечные устройства для выявления угроз и осуществления реагирования на эти угрозы.
- Сбор инвентарной информации, сведений об уязвимом ПО, телеметрии и событий на конечных устройствах.

Технические характеристики

- Функционирование системы на базе технологии Kubernetes с поддержкой Linux-контейнеров версии не ниже 1.24.
- Создание пользовательских элементов экспертизы в рамках единой схемы описания объектов RObject, с использованием языка VRL.

- База данных хранения событий на базе технологии распределенной СУБД ClickHouse.
- Операционная база данных на базе технологии СУБД PostgreSQL.
- Хранение данных в горячем или холодном хранилище.
- Предустановленные виджеты и дашборды для визуального анализа данных.
- Отчеты для анализа и представления данных, собранных системой.
- Поддерживаемые среды функционирования:
  - ALT Linux 10c и выше;
  - Astra Linux SE 1.7.2 и выше;
  - РЕД ОС 7.3c и выше.
- Поддерживаемые версии браузеров:
  - Google Chrome 97 и выше;
  - Microsoft Edge 97 и выше;
  - Mozilla Firefox 100 и выше.

  Рекомендуется использовать наиболее актуальные версии браузеров.

Развертывание системы

В данном разделе приведены инструкции по настройке окружения, установке компонентов системы и их первоначальной настройке, а также аппаратные и программные требования к серверам.

Чтобы установить систему, выполните следующие действия:

1. Убедитесь, что окружение, в котором будет установлена система, соответствует техническим требованиям.
2. Разверните кластер Kubernetes, в котором будет работать система.
3. Установите СУБД ClickHouse и PostgreSQL.

Для повышения стабильности работы системы и удобства обслуживания СУБД рекомендуется устанавливать ClickHouse и PostgreSQL на выделенных серверах.

4. Выполните подготовительные действия перед установкой системы: настройте сетевой доступ к системе и кластеру, в котором она развернута, и скачайте дистрибутив.
5. Установите систему.
6. Если требуется, установите сателлиты — вспомогательные экземпляры системы, развернутые на дополнительных удаленных кластерах.

Чтобы обновить систему, обратитесь к разделу Обновление системы.

Если в процессе установки или обновления системы возникнут неполадки, обратитесь к разделу Устранение проблем при установке или обновлении.

В разделе Установка и удаление агентов приведены инструкции по установке, настройке и удалению агентов, собирающих события на рабочих станциях.

Чтобы удалить систему, обратитесь к разделу Удаление системы.

Технические требования

В данном разделе представлены минимальные технические требования для установки и функционирования системы, в том числе для ее основного экземпляра и сателлитов.

- Программные требования
- Аппаратные требования

Программные требования

Поддерживаемые операционные системы

Для основного экземпляра системы:

- ALT Linux 10c и выше;
- Astra Linux SE 1.7.2 и выше;

- РЕД ОС 7.3c и выше.

В случае сателлитов требования к операционной системе узлов кластера не предъявляются.

Требования к Kubernetes

- Минимальная версия Kubernetes: 1.24.
- Поддерживаемая среда выполнения контейнеров (CRI): containerd, CRI-O.

Выбор среды выполнения имеет значение для автономной установки без Docker-registry в инфраструктуре.

Требования к СУБД

PostgreSQL

Для работы системы требуется реляционная СУБД:

- рекомендуется PostgreSQL версии 15 или выше;
- допустимо использование совместимых аналогов: Postgres Pro или Jatoba J4.

PostgreSQL может быть развернута **в кластере Kubernetes** или **на выделенном сервере**. Настройка подключения к установленному экземпляру PostgreSQL производится на этапе **PostgreSQL setup** установки системы.

Установка PostgreSQL на выделенный сервер не является обязательной, однако рекомендуется для повышения стабильности работы системы и удобства обслуживания СУБД.

ClickHouse

Для работы системы требуется СУБД ClickHouse.

Поддерживаемые версии ClickHouse: 24.7 и выше.

Рекомендуется устанавливать ClickHouse максимальной поддерживаемой версии. Более новые версии ClickHouse обладают большим объемом функциональности и отличаются более высокой производительностью.

Для корректной работы системы необходимо использовать ClickHouse поддерживаемой версии. Установка неподдерживаемых версий может привести к непредвиденным последствиям.

ClickHouse может быть развернута **в кластере Kubernetes** или **на выделенном сервере**. Настройка подключения к установленному экземпляру ClickHouse производится на этапе **ClickHouse setup** установки системы.

Установка ClickHouse на выделенный сервер не является обязательной, однако рекомендуется для повышения стабильности работы системы и удобства обслуживания СУБД.

Системные утилиты

Утилита	Минимальная версия	Наличие на узлах Kubernetes и АРМ администратора
bash	4.0	требуется
gunzip	любая	требуется
sed	любая	требуется только на АРМ администратора
tar	любая	требуется только на АРМ администратора
curl	любая	требуется только на АРМ администратора
kubectl	1.24	требуется только на АРМ администратора
ansible	2.9	требуется
python	3.8	требуется

АРМ администратора — автоматизированное рабочее место администратора.

Поддерживаемые версии браузеров на рабочих станциях

- Google Chrome 97 и выше;
- Microsoft Edge 97 и выше;
- Mozilla Firefox 100 и выше.

Рекомендуется использовать наиболее актуальные версии браузеров.

Аппаратные требования

Требования для работы установщика

Для основного экземпляра системы: от 20 ГБ свободного места на диске.

Для сателлитов: от 10 ГБ свободного места на диске.

Требования для узлов кластера Kubernetes

Планирование ресурсов для развертывания кластера Kubernetes должно выполняться исходя из ожидаемой нагрузки и требуемого срока хранения данных в системе.

Рекомендуется использовать:

- процессоры с тактовой частотой не менее 2 ГГц, со всеми производительными ядрами;
- оперативную память DDR4 с тактовой частотой не менее 3200 МГц в двухканальном или четырехканальном режиме.

При распределенной установке необходимо обеспечить сетевое взаимодействие между узлами кластера с пропускной способностью не менее 10 Гбит/с.

На узлах кластера следует использовать исключительно **твердотельные накопители (SSD) со скоростью чтения не менее 550 Мбит/с**. Рекомендуется

объединить SSD в массив RAID 1.

Требования для основного экземпляра системы

Рекомендации по разметке дискового пространства

Узел	Раздел	Назначение раздела	Рекомендуемое свободное пространство
Master	`/var/lib`	Хранение образов Docker	Не менее 30 ГБ
	`/opt/r-vision`	Хранение файлов установщика	20 ГБ
Worker	`/var/lib`	Хранение образов Docker	Не менее 50 ГБ
	`/opt/local-path-provisioner`	Данные ClickHouse	Не менее 30 ГБ
		Служебные данные	Не менее 100 ГБ

Требования к управляющему узлу

Характеристика	Требования
Процессор	8vCPU 2 ГГц
Память	16 ГБ
SSD	100 ГБ
Сетевое соединение	Не менее 100 Мбит/с

Требования к рабочему узлу

Количество активов	Требования	Сетевое соединение
10 000	CPU: 24 x 2 ГГц RAM: 32 ГБ SSD: 300 ГБ	Не менее 100 Мбит/с
20 000	CPU: 32 x 2 ГГц RAM: 48 ГБ SSD: 300 ГБ	Не менее 100 Мбит/с
50 000	CPU: 48 x 2 ГГц RAM: 64 ГБ SSD: 450 ГБ	Не менее 300 Мбит/с

Количество активов	Требования	Сетевое соединение
100 000 и более	CPU: 64 x 2 ГГц RAM: 96 ГБ SSD: 450 ГБ	Не менее 300 Мбит/с

Требования для сателлитов

Характеристика	Требования
Процессор	8vCPU 2 ГГц
Память	16 ГБ
SSD	100 ГБ
Сетевое соединение	Не менее 100 Мбит/с

Требования к серверу БД

Количество активов	Требования	Сетевое соединение
10 000	CPU: 6 x 2 ГГц RAM: 12 ГБ SSD: 500 ГБ	Не менее 100 Мбит/с
20 000	CPU: 8 x 2 ГГц RAM: 16 ГБ SSD: 500 ГБ	Не менее 100 Мбит/с
50 000	CPU: 16 x 2 ГГц RAM: 32 ГБ SSD: 750 ГБ	Не менее 100 Мбит/с
100 000	CPU: 24 x 2 ГГц RAM: 56 ГБ SSD: 1 ТБ	Не менее 100 Мбит/с
100 000 и более	CPU: 48 x 2 ГГц RAM: 72 ГБ SSD: 1,5 ТБ	Не менее 100 Мбит/с

Развертывание кластера Kubernetes

До установки системы необходимо развернуть кластер Kubernetes.

Это руководство описывает процесс офлайн-установки кластера Kubernetes в окружении РЕД ОС с использованием автономного установщика на базе Kubespray. Для установки в окружении других ОС с использованием подключения к Интернету обратитесь к документации производителей ОС, например:

- Руководство администратора РЕД ОС 7.3.
- Руководство администратора Альт 8 СП.

Подготовка зависимостей

Необходимые архивы

Для установки потребуется архив kubespray-installer-redos.tar.gz для РЕД ОС. Его можно скачать из файлового хранилища R-Vision.

Если у вас есть скачанный установщик системы, архив можно загрузить из его меню. Для этого:

1. Запустите установщик системы:

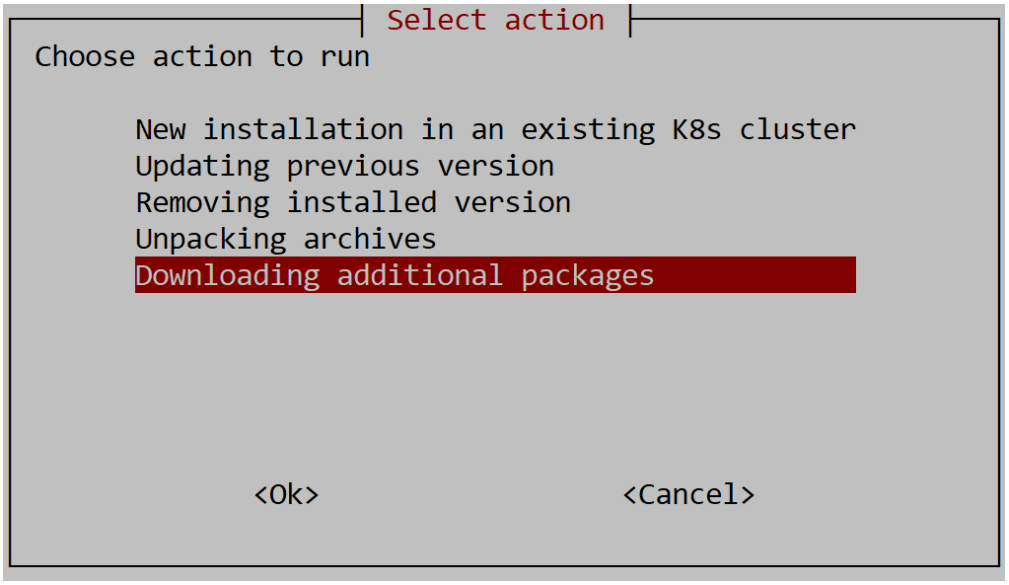
   ```
   bash ./evo.platform_<version>.run
   ```

   Требуемая версия bash: 4.0 или выше.

   Или сделайте файл установщика исполняемым и запустите непосредственно:

   ```
   chmod +x ./evo.platform_<version>.run
   ./evo.platform_<version>.run
   ```

2. Выберите режим работы **Downloading additional packages**.



3. Загрузите дополнительный пакет kubespray-installer-redos.tar.gz.

Архив с установочным файлом kubespray-installer-redos.tar.gz необходимо загрузить во временную директорию `/tmp`.

Распаковка архива

Выполните следующие шаги на управляющем узле или машине администратора:

1. Загрузите файл kubespray-installer-redos.tar.gz в директорию на сервер, на котором будет производиться установка:

   Пример 1. Для Windows

   ```
   pscp "путь\к\файлу\kubespray-installer-redos.tar.gz" user@server:<installation_dir>
   ```

   Пример 2. Для Linux или macOS

   ```
   scp путь\к\файлу\kubespray-installer-redos.tar.gz user@server:<installation_dir>
   ```

2. Распакуйте архив с установочными файлами. Используйте следующие команды:

   ```
   cd <installation_dir>
   tar xvfz kubespray-installer-redos.tar.gz
   ```

Здесь:

- `<installation_dir>` — директория, в которой будет производиться установка (будет размещен файл дистрибутива evo-<номер версии>.run).

Установка кластера Kubernetes

Чтобы избежать ошибок разрешения имен, перед развертыванием кластера

отключите использование IPv6, выполнив следующие действия на каждом узле кластера:

1. Откройте файл /etc/sysctl.conf.
2. Добавьте в файл следующие строки:

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.local.disable_ipv6 = 1
```

3. Сохраните файл.
4. Примените изменения с помощью следующей команды:

```
sysctl -p
```

Настройка SSH-доступа

Проверьте возможность подключения по SSH-ключу к каждому узлу кластера:

```
ssh <FQDN_addr>
```

Здесь:

- `<FQDN_addr>` — FQDN-адрес текущего узла.

Чтобы узнать FQDN-адрес текущего узла, выполните команду: `hostname -f`.

Если подключение по SSH-ключу не настроено, сгенерируйте ключи и добавьте открытый ключ на каждый узел:

```
ssh-keygen
ssh-copy-id <user>@<FQDN_addr>
```

Установка утилит

На каждом узле установите утилиты, требуемые для подключения к кластеру:

```
sudo dnf install bash-completion vim telnet chrony jq htop tcpdump zip unzip rsync
ebtables ethtool conntrack -y
```

Синхронизация времени с NTP-сервером

Настройте синхронизацию времени с корпоративным NTP-сервером, выполнив следующие команды на каждом узле кластера:

```
sudo timedatectl set-ntp true
sudo sed -i "/^server /d" /etc/chrony.conf
sudo bash -c 'echo "server <time_server>" >> /etc/chrony.conf'
sudo systemctl enable --now chronyd
sudo systemctl restart chronyd
sudo systemctl status chronyd
chronyc sources
```

Здесь:

- `<time_server>` — адрес корпоративного сервера времени (например, `ntp.rvision.local`).

Запуск скрипта установки

Скрипт принимает следующие переменные окружения:

Переменная	Описание
`ASK_PASS`	Запрос пароля для подключения к узлам по SSH.
`ANSIBLE_USER`	Пользователь для подключения к узлам по SSH.
`ANSIBLE_PASSWORD`	Пароль для подключения к узлам по SSH.
`ASK_BECOME_PASS`	Запрос пароля для повышения до sudo.
`VERBOSE`	Уровень детализации логирования (0-4). По умолчанию установлен минимальный уровень детализации (0).

Пример запуска с переменными окружения:

```
ANSIBLE_USER=username ANSIBLE_PASSWORD=password ./install.sh
```

По умолчанию скрипт установки запускается в интерактивном режиме, при котором отдельно запрашиваются FQDN-адреса для каждого узла кластера и путь до директории с данными containerd. Чтобы отключить интерактивность скрипта установки, необходимо сразу передать эти данные при его запуске с помощью следующих переменных окружения:

Переменная	Описание
`MASTER_NODES`	Список FQDN-адресов управляющих узлов через пробел.
`WORKER_NODES`	Список FQDN-адресов рабочих узлов через пробел.
`CONTAINERD_ROOT`	Путь к директории с данными containerd (по умолчанию `/var/lib/containerd`).

Пример запуска в неинтерактивном режиме:

```
ANSIBLE_USER=username ANSIBLE_PASSWORD=password MASTER_NODES="m01.example.com"
WORKER_NODES="w01.example.com w02.example.com" CONTAINERD_ROOT="/var/lib/containerd"
```

```
./install.sh
```

Если пользователь не имеет права записи в корневой каталог, установщик необходимо запускать через sudo:

```
sudo ANSIBLE_USER=username ANSIBLE_PASSWORD=password MASTER_NODES="m01.example.com"
WORKER_NODES="w01.example.com w02.example.com" CONTAINERD_ROOT="/var/lib/containerd"
./install.sh
```

Проверка работоспособности

После завершения установки проверьте статус узлов кластера:

```
kubectl get nodes
```

Если все узлы кластера имеют статус Ready, значит, кластер успешно развернут.

Установка СУБД

В данном разделе описаны действия по установке СУБД, необходимых для работы системы: ClickHouse и PostgreSQL.

ClickHouse и PostgreSQL могут быть развернуты **в кластере Kubernetes** или **на выделенных серверах**.

Если планируется установка ClickHouse и PostgreSQL в кластер Kubernetes, перейдите к разделу Подготовка к установке системы. Кластерная установка СУБД будет произведена автоматически в процессе установки системы.

Иначе, если планируется установка ClickHouse и PostgreSQL на выделенные серверы, ознакомьтесь со следующими разделами:

- Установка ClickHouse на выделенном сервере
- Установка PostgreSQL на выделенном сервере

Установка ClickHouse на выделенном сервере

В этом разделе описан процесс установки СУБД ClickHouse на выделенном сервере.

Поддерживаемые версии ClickHouse: 24.7 и выше.

Рекомендуется устанавливать ClickHouse максимальной поддерживаемой версии. Более новые версии ClickHouse обладают большим объемом функциональности и отличаются более высокой производительностью.

Для корректной работы системы необходимо использовать ClickHouse поддерживаемой версии. Установка неподдерживаемых версий может привести к

непредвиденным последствиям.

Если у вас еще не развернут кластер ClickHouse, воспользуйтесь Ansible-плейбуком для автономной установки и настройки кластера с нуля.

Иначе, если у вас уже есть установленный кластер ClickHouse, настройте его конфигурацию для работы с системой.

Настройка нового кластера ClickHouse

Данная инструкция описывает установку и настройку кластера ClickHouse с нуля. При необходимости настроить уже существующий кластер ClickHouse, обратитесь к разделу Настройка существующего кластера ClickHouse.

Инструкция предлагает автоматизированный способ установки ClickHouse на РЕД ОС 7.3 с помощью Ansible-плейбука. Плейбук содержится в архиве clickhouse_offline-playbook.zip, который поставляется сетевыми инженерами по запросу.

Вы можете установить ClickHouse самостоятельно, пользуясь инструкцией из официальной документации ClickHouse.

Если вы установили ClickHouse самостоятельно, перейдите к разделу Установка PostgreSQL на выделенном сервере.

Установка ClickHouse

Действия выполняются на управляющем узле кластера.

Чтобы запустить установку:

1. Скачайте и распакуйте архив clickhouse_offline-playbook.zip на управляющем узле кластера Kubernetes.
2. Перейдите в директорию распакованного архива и измените файл hosts.ini:
   - В разделе `[clickhouse]` укажите полные доменные адреса (FQDN) машин, на которых будет развернут кластер ClickHouse. Рядом с каждым адресом необходимо также указать ее идентификатор `id`. Рекомендуется указывать `id` по порядку начиная с "1".
   - В разделе `[clickhouse:vars]` укажите данные учетной записи с правами администратора для работы под суперпользователем. Если к узлам кластера настроен SSH-доступ, то удалите строку `ansible_password`.

▼ *Пример содержимого файла hosts.ini*

```
[clickhouse]
localhost.localdomain1 id=1
localhost.localdomain2 id=2
[clickhouse:vars]
ansible_user=user
ansible_password=password
```

3. Перейдите в директорию `roles/clickhouse-role/defaults` в распакованном архиве и настройте параметры кластера в файле main.yml. Перечень параметров приведен в разделе Параметры кластера ClickHouse данной статьи.

▼ *Пример содержимого файла main.yml*

```
clickhouse_version: "24.3.1.2672"
# Учетные данные для подключения к СУБД.
clickhouse_user: user
clickhouse_password: password
# Переменные для файлов конфигурации ClickHouse и Keeper. Оставить без изменений.
clickhouse_servers: "{{ groups['clickhouse'] }}"
# API-порт для взаимодействия системы и ClickHouse.
grpc_port: 9101
# HTTP-порт для взаимодействия системы и ClickHouse. По умолчанию порт 8123,
можно скорректировать (будет запрошен установщиком системы).
http_port: 8123
# Пароли пользователей owner, writer, reader.
owner_password: OwnerPassword
writer_password: WriterPassword
reader_password: ReaderPassword

# Параметры сервера ClickHouse.
background_buffer_flush_schedule_pool_size: 6
background_common_pool_size: 4
background_distributed_schedule_pool_size: 8
background_fetches_pool_size: 4
background_message_broker_schedule_pool_size: 8
background_move_pool_size: 4
background_pool_size: 8
background_merges_mutations_concurrency_ratio: 4
background_schedule_pool_size: 64
mark_cache_size: 5368709120
max_concurrent_queries: 0
max_server_memory_usage: 0
# Параметры таблиц MergeTree.
max_bytes_to_merge_at_max_space_in_pool: 2147483648
merge_max_block_size: 2048
number_of_free_entries_in_pool_to_lower_max_size_of_merge: 4
max_suspicious_broken_parts: 500

# Директории хранения данных.
hot_volume_path: /var/lib/clickhouse/hot-1/
cold_volume_path: /var/lib/clickhouse/cold-1/
```

4. Перейдите в директорию распакованного архива и запустите плейбук установки:

```
ansible-playbook main.yml -i hosts.ini -K -vvv
```

Плейбук запросит пароль для повышения привилегий учетной записи до суперпользователя, после чего начнется установка.

Параметры кластера ClickHouse

Учетные данные для подключения к СУБД

В целях повышения безопасности, для подключения к СУБД ClickHouse и непосредственного управления самой базой данных, плейбук создает специального пользователя. Учетные данные этого пользователя задаются в следующих параметрах:

- `clickhouse_user` — имя пользователя.
- `clickhouse_password` — пароль пользователя. Пароль может содержать только строчные латинские буквы (`a–z`), заглавные латинские буквы (`A–Z`) и цифры (`0–9`).

Порты для взаимодействия системы и ClickHouse

API-порт для взаимодействия системы и ClickHouse задается в параметре `grpc_port`.

По умолчанию установлен порт 9100. Порт 9100 используется для приложения Node Exporter на рабочих и управляющих узлах кластера, поэтому рекомендуется задать другой порт, например 9101.

Для взаимодействия системы и ClickHouse по протоколу HTTP по умолчанию используется порт 8123. Его можно изменить в параметре `http_port`.

Пароли пользователей owner, writer, reader

Для работы системы необходимо, чтобы в ClickHouse были настроены три пользователя с разным набором прав:

- **owner** — пользователь с расширенным набором прав. Используется для создания, обновления и удаления таблиц (хранилищ событий), а также получения служебной информации.
- **writer** — пользователь с правами на запись. Используется в коллекторах для записи событий в ClickHouse.
- **reader** — пользователь с правами только на чтение. Используется для выполнения поисковых запросов.

Пароли для этих пользователей задаются в параметрах `owner_password`, `writer_password` и `reader_password` соответственно. Пароли должны отвечать следующим требованиям:

- Пароль может содержать только строчные латинские буквы (`a–z`), заглавные латинские буквы (`A–Z`) и цифры (`0–9`).
- Минимальная длина пароля: 8 знаков.

Эти пароли необходимо использовать на этапе **ClickHouse setup** установки системы.

Параметры ClickHouse

В зависимости от планируемой нагрузки на систему и доступных вычислительных ресурсов вы

можете настроить параметры сервера ClickHouse и таблиц MergeTree, перечисленные в разделе Справка о параметрах ClickHouse.

▼ *Пример настройки параметров ClickHouse*

```
background_buffer_flush_schedule_pool_size: 6
background_common_pool_size: 4
background_distributed_schedule_pool_size: 8
background_fetches_pool_size: 4
background_message_broker_schedule_pool_size: 8
background_move_pool_size: 4
background_pool_size: 8
background_merges_mutations_concurrency_ratio: 4
background_schedule_pool_size: 64
mark_cache_size: 5368709120
max_concurrent_queries: 0
max_server_memory_usage: 0
```

▼ *Пример настройки параметров таблиц MergeTree*

```
max_bytes_to_merge_at_max_space_in_pool: 2147483648
merge_max_block_size: 2048
number_of_free_entries_in_pool_to_lower_max_size_of_merge: 4
max_suspicious_broken_parts: 500
```

Директории хранения данных

Директории хранения данных ClickHouse задаются в следующих параметрах:

- `hot_volume_path` — директория для хранения часто используемых данных (горячее хранение);
- `cold_volume_path` — директории для хранения редко используемых данных (холодное хранение).

Если хранение событий планируется в корневую директорию ClickHouse, то оставьте настройки по умолчанию.

Пример 3. Пример настройки директорий хранения

```
hot_volume_path: /var/lib/clickhouse/hot-1/
cold_volume_path: /var/lib/clickhouse/cold-1/
```

Если планируется хранить события в отдельной директории, то:

1. Создайте необходимые директории под горячее и холодное хранение данных с помощью команды:

   ```
   mkdir путь/до/директории
   ```

2. Выдайте права на созданные директории группе пользователей clickhouse с помощью команды:

```
chown -R clickhouse:clickhouse путь/до/директории
```

3. Укажите пути до созданных директорий в параметрах `hot_volume_path` и `cold_volume_path` в файле main.yml.

Пример настройки отдельных директорий

Например, необходимо хранить события ClickHouse в директориях `/storage/hot` (горячее хранение) и `/storage/cold` (холодное хранение).

Чтобы настроить хранение в этих директориях:

1. Создайте директории с помощью команд:

```
mkdir /storage/hot
mkdir /storage/cold
```

2. Выдайте права на директории группе пользователей clickhouse с помощью команд:

```
chown -R clickhouse:clickhouse /storage/hot
chown -R clickhouse:clickhouse /storage/cold
```

Если в директории `/storage` планируется хранение только данных событий ClickHouse, то, вместо выдачи прав на каждую поддиректорию по отдельности, вы можете выдать права сразу на всю директорию `/storage`:

```
chown -R clickhouse:clickhouse /storage
```

3. Укажите пути до директорий в параметрах `hot_volume_path` и `cold_volume_path` в файле main.yml:

```
hot_volume_path: /storage/hot/
cold_volume_path: /storage/cold/
```

Настройка существующего кластера ClickHouse

Данная инструкция описывает настройку уже существующего кластера ClickHouse. Для установки и настройки кластера с нуля обратитесь к разделу Настройка нового кластера ClickHouse.

Настройка существующего кластера ClickHouse состоит из следующих этапов:

1. Настройка конфигурации кластера ClickHouse
2. Запуск кластера ClickHouse
3. Создание пользователей ClickHouse

Настройка конфигурации кластера ClickHouse

Действия выполняются на каждом узле кластера.

Чтобы настроить конфигурацию кластера ClickHouse, на каждом узле кластера перейдите в директорию `config.d` в директории установки сервера ClickHouse и настройте параметры кластера в конфигурационном файле config.xml.

▼ *Пример содержимого файла config.xml*

```
<clickhouse>
    <access_control_improvements>

<users_without_row_policies_can_read_rows>true</users_without_row_policies_can_read_
rows>

<on_cluster_queries_require_cluster_grant>false</on_cluster_queries_require_cluster_
grant>

<select_from_system_db_requires_grant>false</select_from_system_db_requires_grant>

<select_from_information_schema_requires_grant>false</select_from_information_schema
_requires_grant>

<settings_constraints_replace_previous>false</settings_constraints_replace_previous>
        <role_cache_expiration_time_seconds>600</role_cache_expiration_time_seconds>
    </access_control_improvements>
    <jdbc_bridge>
        <host>127.0.0.1</host>
        <port>9019</port>
    </jdbc_bridge>
    <!-- API-порт для взаимодействия системы и ClickHouse. -->
    <grpc_port>9101</grpc_port>
    <!-- HTTP-порт для взаимодействия системы и ClickHouse. -->
    <http_port>8123</http_port>
    <allow_plaintext_password>1</allow_plaintext_password>
    <allow_no_password>1</allow_no_password>
    <allow_implicit_no_password>1</allow_implicit_no_password>
    <timezone>UTC</timezone>
    <interserver_http_host>localhost</interserver_http_host>

<database_atomic_delay_before_drop_table_sec>10</database_atomic_delay_before_drop_t
able_sec>
    <!-- Шардирование и репликация узлов кластера. -->
    <remote_servers>
        <default>
            <shard>
                <internal_replication>true</internal_replication>
                <weight>1</weight>
                <replica>
                    <host>IP или FQDN узла 1</host>
```

```
                    <port>9000</port>
                    <user>writer</user>
                    <password>writer</password>
                </replica>
            </shard>
            <shard>
                <internal_replication>true</internal_replication>
                <weight>1</weight>
                <replica>
                    <host>IP или FQDN узла 2</host>
                    <port>9000</port>
                    <user>writer</user>
                    <password>writer</password>
                </replica>
            </shard>
        </default>
    </remote_servers>
    <macros>
        <cluster>default</cluster>
        <replica>IP или FQDN текущего узла</replica>
        <shard>Имя текущего узла</shard>
        <layer>clickhouse</layer>
    </macros>
    <prometheus>
        <endpoint>/metrics</endpoint>
        <port>9363</port>
        <metrics>true</metrics>
        <events>true</events>
        <asynchronous_metrics>true</asynchronous_metrics>
    </prometheus>
    <listen_host>::</listen_host>
    <!-- Параметры сервера ClickHouse. -->

<background_buffer_flush_schedule_pool_size>6</background_buffer_flush_schedule_pool
_size>
    <background_common_pool_size>4</background_common_pool_size>

<background_distributed_schedule_pool_size>8</background_distributed_schedule_pool_s
ize>
    <background_fetches_pool_size>4</background_fetches_pool_size>

<background_message_broker_schedule_pool_size>8</background_message_broker_schedule_
pool_size>
    <background_move_pool_size>4</background_move_pool_size>
    <background_pool_size>8</background_pool_size>

<background_merges_mutations_concurrency_ratio>4</background_merges_mutations_concur
rency_ratio>
    <background_schedule_pool_size>64</background_schedule_pool_size>
    <mark_cache_size>5368709120</mark_cache_size>
    <max_concurrent_queries>0</max_concurrent_queries>
```

```
    <max_server_memory_usage>0</max_server_memory_usage>
    <!-- Параметры таблиц MergeTree. -->
    <merge_tree>

<max_bytes_to_merge_at_max_space_in_pool>2147483648</max_bytes_to_merge_at_max_space
_in_pool>
        <merge_max_block_size>2048</merge_max_block_size>

<number_of_free_entries_in_pool_to_lower_max_size_of_merge>4</number_of_free_entries
_in_pool_to_lower_max_size_of_merge>
        <max_suspicious_broken_parts>500</max_suspicious_broken_parts>
    </merge_tree>
    <!-- Директории хранения данных. -->
    <storage_configuration>
        <disks>
            <hot_disk_1>
                <path>/var/lib/clickhouse/hot-1/</path>
            </hot_disk_1>
            <cold_disk_1>
                <path>/var/lib/clickhouse/cold-1/</path>
            </cold_disk_1>
        </disks>
        <policies>
            <siem_storage_policy>
                <volumes>
                    <hot_volume>
                        <disk>hot_disk_1</disk>
                    </hot_volume>
                    <cold_volume>
                        <disk>cold_disk_1</disk>
                    </cold_volume>
                </volumes>
            </siem_storage_policy>
        </policies>
    </storage_configuration>
    <users>
        <root>
            <access_management>1</access_management>
        </root>
    </users>
    <!-- Настройка ClickHouse Keeper. -->
    <keeper_server>
        <tcp_port>9181</tcp_port>
        <server_id>ID Текущего узла</server_id>
        <log_storage_path>/var/lib/clickhouse/coordination/log</log_storage_path>

<snapshot_storage_path>/var/lib/clickhouse/coordination/snapshots</snapshot_storage_
path>
        <coordination_settings>
            <operation_timeout_ms>10000</operation_timeout_ms>
            <session_timeout_ms>30000</session_timeout_ms>
```

```
            <raft_logs_level>trace</raft_logs_level>
        </coordination_settings>
        <raft_configuration>
            <server>
                <id>ID текущего узла</id>
                <hostname>FQDN текущего узла</hostname>
                <port>9234</port>
            </server>
        </raft_configuration>
    </keeper_server>
    <distributed_ddl>
        <path>/var/lib/clickhouse/task_queue/ddl</path>
    </distributed_ddl>
    <profiles>
        <default>
            <allow_suspicious_ttl_expressions>1</allow_suspicious_ttl_expressions>
        </default>
        <readonly>
            <readonly>1</readonly>
        </readonly>
    </profiles>
</clickhouse>
```

Порты для взаимодействия системы и ClickHouse

API-порт для взаимодействия системы и ClickHouse задается в параметре `grpc_port`.

По умолчанию установлен порт 9100. Порт 9100 используется для приложения Node Exporter на рабочих и управляющих узлах кластера, поэтому рекомендуется задать другой порт, например 9101.

Для взаимодействия системы и ClickHouse по протоколу HTTP по умолчанию используется порт 8123. Его можно изменить в параметре `http_port`.

Шардирование и репликация узлов кластера

Шардирование позволяет разбивать данные БД на части и размещать их на разных узлах кластера (шардах). Репликация предназначена для управления копиями данных (репликами) в БД и их синхронизации. Эти механизмы обеспечивают отказоустойчивость работы кластера и повышают надежность хранения данных.

Шардирование и репликация узлов задаются в блоке `remote_servers`.

▼ *Пример настройки шардирования и репликации*

```
<remote_servers>
    <default>
        <!-- Параметры шарда 1. -->
        <shard>
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
```

```
                <host>IP-адрес или FQDN узла 1</host>
                <port>9000</port>
                <user>writer</user>
                <password>writer</password>
            </replica>
        </shard>
        <!-- Параметры шарда 2. -->
        <shard>
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
                <host>IP-адрес или FQDN узла 2</host>
                <port>9000</port>
                <user>writer</user>
                <password>writer</password>
            </replica>
        </shard>
        <!-- Здесь пропущены параметры промежуточных шардов. -->
        <!-- Параметры шарда N. -->
        <shard>
            <internal_replication>true</internal_replication>
            <weight>1</weight>
            <replica>
                <host>IP-адрес или FQDN узла N</host>
                <port>9000</port>
                <user>writer</user>
                <password>writer</password>
            </replica>
        </shard>
    </default>
</remote_servers>
<macros>
    <cluster>default</cluster>
    <replica>IP-адрес или FQDN текущего узла</replica>
    <shard>Имя текущего узла</shard>
    <layer>clickhouse</layer>
</macros>
```

Если система работает только на одном узле, то настройка выглядит следующим образом:

▼ *Пример настройки шардирования для системы на одном узле*

```
<remote_servers>
    <default>
        <shard>
            <replica>
                <host>localhost</host>
                <port>9000</port>
                <user>writer</user>
                <password>writer</password>
            </replica>
```

```
        </shard>
    </default>
</remote_servers>
<macros>
    <cluster>default</cluster>
    <shard>1</shard>
    <replica>Имя узла</replica>
</macros>
```

Параметры ClickHouse

В зависимости от планируемой нагрузки на систему и доступных вычислительных ресурсов вы можете настроить параметры сервера ClickHouse и таблиц MergeTree, перечисленные в разделе Справка о параметрах ClickHouse.

▼ *Пример настройки параметров ClickHouse*

```
<background_buffer_flush_schedule_pool_size>6</background_buffer_flush_schedule_pool
_size>
<background_common_pool_size>4</background_common_pool_size>
<background_distributed_schedule_pool_size>8</background_distributed_schedule_pool_s
ize>
<background_fetches_pool_size>4</background_fetches_pool_size>
<background_message_broker_schedule_pool_size>8</background_message_broker_schedule_
pool_size>
<background_move_pool_size>4</background_move_pool_size>
<background_pool_size>8</background_pool_size>
<background_merges_mutations_concurrency_ratio>4</background_merges_mutations_concur
rency_ratio>
<background_schedule_pool_size>64</background_schedule_pool_size>
<mark_cache_size>5368709120</mark_cache_size>
<max_concurrent_queries>0</max_concurrent_queries>
<max_server_memory_usage>0</max_server_memory_usage>
```

▼ *Пример настройки параметров таблиц MergeTree*

```
<merge_tree>

<max_bytes_to_merge_at_max_space_in_pool>2147483648</max_bytes_to_merge_at_max_space
_in_pool>
    <merge_max_block_size>2048</merge_max_block_size>

<number_of_free_entries_in_pool_to_lower_max_size_of_merge>4</number_of_free_entries
_in_pool_to_lower_max_size_of_merge>
    <max_suspicious_broken_parts>500</max_suspicious_broken_parts>
</merge_tree>
```

Директории хранения данных

Директории хранения данных ClickHouse задаются в блоке `storage_configuration`:

- В блоке `disks` задаются диски, которые используются томами для хранения данных.
- В блоке `volumes` задаются тома для хранения данных:
  - `hot_volume` — том для хранения часто используемых данных (горячее хранение);
  - `cold_volume` — том для хранения редко используемых данных (холодное хранение).

Если хранение событий планируется в корневой директории ClickHouse, то используйте следующие настройки:

▼ *Настройки хранения событий в корневой директории ClickHouse*

```
<storage_configuration>
    <disks>
        <!-- Диски, используемые для хранения данных. -->
        <hot_disk_1>
            <!-- Пользовательское название диска. -->
            <path>/var/lib/clickhouse/hot-1/</path>
            <!-- Директория хранения данных. -->
        </hot_disk_1>
        <cold_disk_1>
            <path>/var/lib/clickhouse/cold-1/</path>
        </cold_disk_1>
    </disks>
    <policies>
        <!-- Политики хранения, задаваемые при создании таблицы. -->
        <siem_storage_policy>
            <!-- Название политики. -->
            <volumes>
                <!-- Тома хранения. Приоритет записи определяется порядком указания
томов. -->
                <hot_volume>
                    <!-- Том горячего хранения данных. -->
                    <disk>hot_disk_1</disk>
                    <!-- Диск из раздела disks. -->
                </hot_volume>
                <cold_volume>
                    <!-- Том холодного хранения данных. -->
                    <disk>cold_disk_1</disk>
                </cold_volume>
            </volumes>
        </siem_storage_policy>
    </policies>
</storage_configuration>
```

Если планируется хранить события в отдельной директории, то:

1. Создайте необходимые директории под горячее и холодное хранение данных с помощью

команды:

```
mkdir путь/до/директории
```

2. Выдайте права на созданные директории группе пользователей clickhouse с помощью команды:

```
chown -R clickhouse:clickhouse путь/до/директории
```

3. Укажите пути до созданных директорий в `path` блоков `hot_disk_1` и `cold_disk_1`.

Пример настройки отдельных директорий

Например, необходимо хранить события ClickHouse в директориях `/storage/hot` (горячее хранение) и `/storage/cold` (холодное хранение).

Чтобы настроить хранение в этих директориях:

1. Создайте директории с помощью команд:

```
mkdir /storage/hot
mkdir /storage/cold
```

2. Выдайте права на директории группе пользователей clickhouse с помощью команд:

```
chown -R clickhouse:clickhouse /storage/hot
chown -R clickhouse:clickhouse /storage/cold
```

Если в директории `/storage` планируется хранение только данных событий ClickHouse, то, вместо выдачи прав на каждую поддиректорию по отдельности, вы можете выдать права сразу на всю директорию `/storage`:

```
chown -R clickhouse:clickhouse /storage
```

3. Укажите пути до директорий в `path` блоков `hot_disk_1` и `cold_disk_1` в файле config.xml.

▼ *Пример настройки директорий горячего и холодного хранения*

```
<storage_configuration>
    <disks>
        <!-- Диски, используемые для хранения данных. -->
        <hot_disk_1>
            <!-- Пользовательское название диска. -->
            <path>/storage/hot/</path>
            <!-- Директория хранения данных. -->
        </hot_disk_1>
```

```
        <cold_disk_1>
            <path>/storage/cold/</path>
        </cold_disk_1>
    </disks>
    <policies>
        <!-- Политики хранения, задаваемые при создании таблицы. -->
        <siem_storage_policy>
            <!-- Название политики. -->
            <volumes>
                <!-- Тома хранения. Приоритет записи определяется порядком
указания томов. -->
                <hot_volume>
                    <!-- Том горячего хранения данных. -->
                    <disk>hot_disk_1</disk>
                    <!-- Диск из раздела disks. -->
                </hot_volume>
                <cold_volume>
                    <!-- Том холодного хранения данных. -->
                    <disk>cold_disk_1</disk>
                </cold_volume>
            </volumes>
        </siem_storage_policy>
    </policies>
</storage_configuration>
```

Настройка ClickHouse Keeper

ClickHouse Keeper — это служба для управления шардированием и репликацией данных в кластере ClickHouse. Подробную информацию о службе можно найти в официальной документации ClickHouse.

Настройка ClickHouse Keeper выполняется в блоке `keeper_server`.

▼ *Пример настроек ClickHouse Keeper*

```
<keeper_server>
    <tcp_port>9181</tcp_port>
    <server_id>ID текущего узла</server_id>
    <log_storage_path>/var/lib/clickhouse/coordination/log</log_storage_path>

<snapshot_storage_path>/var/lib/clickhouse/coordination/snapshots</snapshot_storage_
path>
    <coordination_settings>
        <operation_timeout_ms>10000</operation_timeout_ms>
        <session_timeout_ms>30000</session_timeout_ms>
        <raft_logs_level>trace</raft_logs_level>
    </coordination_settings>

    <raft_configuration>
        <!-- Параметры узла 1. -->
        <server>
```

```
            <id>ID узла 1</id>
            <hostname>FQDN узла 1</hostname>
            <port>9234</port>
        </server>
        <!-- Параметры узла 2. -->
        <server>
            <id>ID узла 2</id>
            <hostname>FQDN узла 2</hostname>
            <port>9234</port>
        </server>
        <!-- Здесь пропущены параметры промежуточных узлов. -->
        <!-- Параметры узла N. -->
        <server>
            <id>ID узла N</id>
            <hostname>FQDN узла N</hostname>
            <port>9234</port>
        </server>
    </raft_configuration>
</keeper_server>
```

Запуск кластера ClickHouse

Действия выполняются на каждом узле кластера.

Чтобы запустить кластер ClickHouse, необходимо запустить каждый его узел по отдельности:

1. Запустите текущий узел кластера:

   ```
   systemctl start clickhouse-server
   ```

   Ошибки запуска можно посмотреть в файле clickhouse-server.err.log или clickhouse-server.log в директории `/var/log/clickhouse-server/`.

2. Проверьте доступ к текущему узлу:

   ```
   clickhouse-client -q "select version()"
   ```

   Если запуск узла произошел успешно, в консоль выведется установленная версия ClickHouse.

Доступ к другим узлам кластера можно проверить удаленно с помощью команды:

```
clickhouse-client -h <имя узла> -q "select version()"
```

Если запуск узла произошел успешно, в консоль выведется установленная версия ClickHouse.

Создание пользователей ClickHouse

Для работы системы необходимо, чтобы в ClickHouse были настроены три пользователя с разным набором прав:

- **owner** — пользователь с расширенным набором прав. Используется для создания, обновления и удаления таблиц (хранилищ событий), а также получения служебной информации.
- **writer** — пользователь с правами на запись. Используется в коллекторах для записи событий в ClickHouse.
- **reader** — пользователь с правами только на чтение. Используется для выполнения поисковых запросов.

Для создания требуемых пользователей и кластерных вариантов служебных таблиц используйте скрипт ниже. Выполните скрипт на любом экземпляре кластера ClickHouse:

```
-- Создать пользователя owner и выдать ему необходимые права.
CREATE USER IF NOT EXISTS owner ON CLUSTER '{cluster}' IDENTIFIED WITH <OwnerPassword>
BY 'owner';
GRANT ON CLUSTER '{cluster}' ALL on default.* TO 'owner';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'owner';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO 'owner';

-- Создать пользователя writer и выдать ему необходимые права.
CREATE USER IF NOT EXISTS writer ON CLUSTER '{cluster}' IDENTIFIED WITH
<WriterPassword> BY 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' INSERT ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' ALTER UPDATE ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' ALTER DELETE ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'writer';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO 'writer';

-- Создать пользователя reader и выдать ему необходимые права.
CREATE USER IF NOT EXISTS reader ON CLUSTER '{cluster}' IDENTIFIED WITH
<ReaderPassword> BY 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON default.* TO 'reader';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'reader';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'reader';
```

```
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO 'reader';

-- Создать необходимые распределенные таблицы.
CREATE TABLE IF NOT EXISTS default.disks ON CLUSTER '{cluster}' AS system.disks ENGINE
= Distributed('{cluster}', system, disks);
CREATE TABLE IF NOT EXISTS default.storage_policies ON CLUSTER '{cluster}' AS
system.storage_policies ENGINE = Distributed('{cluster}', system, storage_policies);
CREATE TABLE IF NOT EXISTS default.parts ON CLUSTER '{cluster}' AS system.parts ENGINE
= Distributed('{cluster}', system, parts);
CREATE TABLE IF NOT EXISTS default.query_log ON CLUSTER '{cluster}' AS
system.query_log ENGINE = Distributed('{cluster}', system, query_log);
CREATE TABLE IF NOT EXISTS default.columns ON CLUSTER '{cluster}' AS system.columns
ENGINE = Distributed('{cluster}', system, columns);
CREATE TABLE IF NOT EXISTS default.parts_columns ON CLUSTER '{cluster}' AS
system.parts_columns ENGINE = Distributed('{cluster}', system, parts_columns);
```

Здесь `<OwnerPassword>`, `<WriterPassword>` и `<ReaderPassword>` — это пароли пользователей owner, write и reader соответственно. Пароли должны отвечать следующим требованиям:

- Пароль может содержать только строчные латинские буквы (`a–z`), заглавные латинские буквы (`A–Z`) и цифры (`0–9`).
- Минимальная длина пароля: 8 знаков.

Эти пароли необходимо использовать на этапе **ClickHouse setup** установки системы.

Справка о параметрах ClickHouse

Параметры сервера

Подробную информацию о параметрах сервера можно найти в официальной документации ClickHouse.

Вы можете настроить следующие параметры сервера ClickHouse:

- `background_buffer_flush_schedule_pool_size` — количество потоков для выполнения фонового сброса данных.

  Рекомендуемое значение: **(1—2) * количество ядер процессора**.

- `background_common_pool_size` — количество потоков для освобождения памяти.

  Рекомендуемое значение: **(0—1) * количество ядер процессора**.

- `background_distributed_schedule_pool_size` — количество потоков для выполнения фоновых задач.

Рекомендуемое значение: **(1—2) * количество ядер процессора**.

- `background_fetches_pool_size` — количество потоков для репликации данных.

  Рекомендуемое значение: **(0—1) * количество ядер процессора**.

- `background_message_broker_schedule_pool_size` — количество потоков для фонового вывода сообщений.

  Рекомендуемое значение: **(1—2) * количество ядер процессора**.

- `background_move_pool_size` — количество потоков для перемещений данных между дисками.

  Рекомендуемое значение: **(0—1) * количество ядер процессора**.

- `background_pool_size` — количество потоков для слияний и мутаций таблиц.

  Рекомендуемое значение: **(1—2) * количество ядер процессора**.

- `background_merges_mutations_concurrency_ratio` — отношение количества слияний и мутаций, которые могут выполняться одновременно, к количеству потоков `background_pool_size`. Например, если параметр принимает значение 2, а `background_pool_size` — 16, то сервер ClickHouse может одновременно выполнять 32 слияния.
- `background_schedule_pool_size` — количество потоков для выполнения фоновых задач.

  Рекомендуемое значение: **(8—16) * количество ядер процессора**.

- Параметры для ограничения количества ошибочных строк при вставке:

  - `input_format_allow_errors_num` — максимально допустимое количество строк с ошибками при выполнении операции вставки. Если строка содержит ошибку, но значение `input_format_allow_errors_num` не превышено, ClickHouse пропускает эту строку и продолжает выполнение запроса. Значение по умолчанию — 0.
  - `input_format_allow_errors_ratio` — максимально допустимый процент строк с ошибками при выполнении операции вставки. Если строка содержит ошибку, но значение `input_format_allow_errors_ratio` не превышено, ClickHouse пропускает эту строку и продолжает выполнение запроса. Измеряется от 0 до 1. Значение по умолчанию — 0.

  Ошибка выполнения запроса возникает только при превышении одновременно `input_format_allow_errors_num` и `input_format_allow_errors_ratio`.

- `mark_cache_size` — приблизительный размер кэша меток в байтах.

  Рекомендуемое значение: **1—5 ГБ**.

- `max_concurrent_queries` — максимальное количество одновременно обрабатываемых запросов. Если принимает значение "0", то ограничений нет.

  Рекомендуемое значение: **(1—2) * количество ядер процессора**.

- `max_server_memory_usage` — максимальный объем оперативной памяти в байтах, используемой сервером ClickHouse. Если принимает значение "0", то объем памяти определяется

автоматически следующим образом: **memory_amount * max_server_memory_usage_to_ram_ratio**.

Здесь:

- `memory_amount` — общий объем оперативной памяти.
- `max_server_memory_usage_to_ram_ratio` — доля оперативной памяти, используемой сервером ClickHouse. Если принимает значение "0", то сервер может использовать всю оперативную память. По умолчанию параметр не задан и воспринимается системой равным "0.9".

Рекомендуемое значение: **90% от доступного объема оперативной памяти**.

Параметры таблиц MergeTree

Таблицы семейства MergeTree позволяют использовать механизм репликации данных. Подробную информацию о MergeTree-параметрах можно найти в официальной документации ClickHouse.

Вы можете настроить следующие параметры MergeTree-таблиц:

- `max_bytes_to_merge_at_max_space_in_pool` — максимальный общий размер таблиц, которые могут быть слиты, в байтах.

  Рекомендуемое значение: **1—4 ГБ**.

- `merge_max_block_size` — количество строк, которые считываются в память из слитых таблиц.

  Рекомендуемое значение: **1024—4096**.

- `number_of_free_entries_in_pool_to_lower_max_size_of_merge` — количество свободных записей в пуле или реплицированной очереди, при преуменьшении которого уменьшится максимальный размер слияния.

  Рекомендуемое значение: **1—4**.

- `max_suspicious_broken_parts` — максимальное количество поврежденных партиций. При превышении этого значения отключается автоматическое удаление.

Установка PostgreSQL на выделенном сервере

В этом разделе описан процесс установки СУБД PostgreSQL на выделенном сервере.

Требования к PostgreSQL

Для работы системы требуется реляционная СУБД:

- рекомендуется PostgreSQL версии 15 или выше;
- допустимо использование совместимых аналогов: Postgres Pro или Jatoba J4.

Установка PostgreSQL

Загрузите и установите PostgreSQL согласно инструкции с официального сайта.

Настройка PostgreSQL на выделенном сервере

Настройка сетевого интерфейса и порта

Прослушиваемые сервером PostgreSQL сетевой интерфейс и порт задаются в конфигурационном файле `postgresql.conf`.

Расположение файла зависит от используемой операционной системы:

- В дистрибутивах Linux, основанных на Debian, файл `postgresql.conf` находится в директории `/etc/postgresql/<version>/main/`.
- В дистрибутивах Linux, основанных на Red Hat, файл `postgresql.conf` находится в директории `/var/lib/pgsql/<version>/data/`.

Если вы используете иную операционную систему, расположение файла `postgresql.conf` может быть другим.

Чтобы настроить сетевой интерфейс и порт:

1. Откройте файл `postgresql.conf` для редактирования.
2. Задайте в параметре `listen_addresses` IP-адрес сервера PostgreSQL, который система будет использовать для подключения к базе данных.
3. Задайте в параметре `port` порт, который будет прослушивать сервер PostgreSQL в ожидании подключений (по умолчанию — 5432).
4. Сохраните и закройте файл `postgresql.conf`.

Настройка сетевого доступа кластера к PostgreSQL

Разрешения сетевого доступа клиентов к серверу PostgreSQL задаются в конфигурационном файле `pg_hba.conf`. Этот файл расположен в той же директории, что и файл `postgresql.conf`.

Чтобы настроить разрешения сетевого доступа:

1. Откройте файл `pg_hba.conf` для редактирования.
2. Разрешите подключение рабочих узлов кластера Kubernetes, добавив их адреса в виде строк:

   ```
   host all all worker-node-ip1/32 scram-sha-256
   host all all worker-node-ip2/32 scram-sha-256
   host all all worker-node-ipN/32 scram-sha-256
   ```

   где `worker-node-ipX` — IP-адреса рабочих узлов кластера, в котором развернута система. В зависимости от настроек PostgreSQL может использоваться алгоритм хэширования `md5` вместо приведенного в примере `scram-sha-256`.

3. Сохраните и закройте файл `pg_hba.conf`.

Применение параметров

Перезапустите PostgreSQL, выполнив следующую команду:

```
service postgresql restart
```

или

```
systemctl restart postgresql
```

Добавление пользователя PostgreSQL

При установке PostgreSQL создает суперпользователя `postgres`. По умолчанию пароль для его учетной записи не задан. Рекомендуется создавать отдельного пользователя с надежным паролем. Пользователь должен обладать привилегией `CREATEDB`, которая позволяет создавать новые базы данных и становиться их владельцем.

Чтобы добавить пользователя:

1. Переключитесь от имени пользователя `postgres` в командной строке:

   ```
   su postgres
   ```

   или

   ```
   sudo -i -u postgres
   ```

2. Добавьте пользователя, выполнив следующую команду:

   ```
   createuser --createdb --pwprompt evo-user
   ```

   или

   ```
   psql -c "CREATE USER \"evo-user\" WITH CREATEDB PASSWORD 'yourpassword';"
   ```

   где `evo-user` — пример имени нового пользователя (может быть другим), `yourpassword` — пример пароля.

3. Пользователь добавлен. Завершите сеанс пользователя `postgres` командой `exit`.

При эксплуатации системы рекомендуется периодически создавать резервные копии баз данных.

Подготовка к установке системы

В данном разделе описаны подготовительные действия, которые необходимо выполнить перед установкой системы.

Подготовка состоит из следующих этапов:

1. Настройка доступа к кластеру Kubernetes.
2. Настройка сетевого доступа к кластеру системы.
3. Настройка доступа к рабочим узлам кластера.
4. Скачивание архива с файлами для установки и проверка его целостности.

Настройка доступа к кластеру Kubernetes

До начала установки должен быть развернут кластер Kubernetes.

Установка может выполняться как с одного из управляющих узлов кластера Kubernetes, так и с машины администратора кластера. У пользователя, от имени которого будет запускаться установщик, должен быть настроен доступ к кластеру Kubernetes.

Вы можете настроить доступ в конфигурационном файле утилиты kubectl — `$HOME/.kube/config` или задать путь к конфигурации в переменной `KUBECONFIG`:

```
export KUBECONFIG=<path_to_confg>
```

Здесь:

- `<path_to_config>` — путь к конфигурации kubectl.

Проверить доступ можно командой:

```
kubectl get nodes
```

Настройка доменного имени и сетевого адреса

Чтобы настроить сетевой доступ к кластеру системы:

1. Запросите у сетевых инженеров IP-адрес или диапазон IP-адресов, которые будут зарезервированы для использования кластером системы. Убедитесь, что эти адреса доступны для обращений из внешних сетей.
2. Добавьте следующие записи в DNS:
   - **A-запись**: должна указывать на IP-адрес, зарезервированный для основного доступа к кластеру. Пример:

     ```
     evo.company.local
     ```

   - **CNAME-запись**: должна указывать на A-запись и используется для создания псевдонима или для обращения к специфическим службам кластера. Пример:

```
grafana.evo.company.local
```

3. Внесите изменения в DNS через интерфейс управления DNS вашей организации или обратитесь к администратору DNS для добавления указанных записей.

Настройка доступа к рабочим узлам кластера

Для загрузки образов контейнеров на рабочие узлы установщик использует Ansible. Необходимо настроить SSH-доступ с машины установщика до рабочих узлов под единой учетной записью. Поддерживается доступ по ключам или паролю.

Учетная запись должна иметь права суперпользователя или иметь возможность повышения прав до суперпользователя.

Получение дистрибутива

Скачайте архив с установочными файлами evo.platform_<version>.run и проверьте его контрольную сумму. Ссылка на дистрибутив предоставляется по запросу в службу поддержки по адресу support@rvision.ru.

```
# Получение дистрибутива.
curl -O https://files.rvision.ru/platform/<random_part>/evo.platform_<version>.run

# Получение контрольной суммы.
curl -O
https://files.rvision.ru/platform/<random_part>/evo.platform_<version>.run.sha256

# Проверка контрольной суммы.
sha256sum -c evo.platform_<version>.run.sha256

# При успешной проверке контрольной суммы будет выведено следующее сообщение.
evo.platform_<version>.run: OK

# При нарушении контрольной суммы будет выведено следующее сообщение.
evo.platform_<version>.run: FAILED
```

При выявлении нарушения контрольной суммы выполнять установку **недопустимо**.

Необходимо убедиться, что скачивание run-файла дистрибутива было полностью завершено, и повторно проверить контрольную сумму.

При подтверждении нарушения контрольной суммы следует сообщить в техническую поддержку по адресу support@rvision.ru.

Установка системы

Система предназначена для работы в среде Kubernetes и поставляется в виде контейнерных образов

и helm-чартов. В данном разделе приведены инструкции по офлайн-установке системы.

Содержание раздела:

- Пакет установки
- Процесс установки системы

Пакет установки

В данном разделе приведено описание дистрибутива и установщика системы.

Описание дистрибутива

Дистрибутив системы рассчитан на автономную установку в существующий кластер Kubernetes без необходимости доступа к сети Интернет. Дистрибутив представляет собой самораспаковывающийся run-файл (bash-скрипт) с полезной нагрузкой.

Дистрибутив состоит из компонентов установщика и модулей платформы.

Компоненты установщика

- Скрипт установки (bash).
- Скрипт управления экземпляром платформы evoctl.
- Плейбуки Ansible.
- Вспомогательные утилиты (исполняемые файлы: `helm`, `jq`, `yq`).

Модули платформы

- Модуль evo.infra, содержащий следующие компоненты:
  - Инфраструктурные сервисы:
    - мониторинг: kube-prometheus-stack;
    - контроллер: ingress-nginx;
    - провайдеры хранилища: local-path-provisioner и piraeus;
    - логирование: loki и promtail;
    - балансировщик нагрузки: metallb;
    - сервер метрик: metrics-server.
  - Сторонние сервисы:
    - ClickHouse;
    - NATS;
    - Consul;
    - Dkron;
    - MinIO;
    - PostgreSQL.

- Архивы модулей платформы.

Описание установщика

Установщик системы — это самораспаковывающийся run-файл (bash-скрипт) с полезной нагрузкой.

Формат имени файла: `evo.platform_<version>.run`

Для вывода встроенной справки запустите run-файл evo.platform_<version>.run с параметром `-h` или `--help`:

```
./evo.platform_<version>.run --help
```

Если файл не имеет прав на выполнение, установите их с помощью команды `chmod`:

```
sudo chmod +x evo.platform_<version>.run
```

▼ *Результат*

```
Usage: evo.platform_<version>.run [OPTIONS]

Self-extracting installer for R-Vision EVO Platform, version: <version>.

Available main actions:
  - new installation;
  - upgrading existing installation;
  - removing product;
  - downloading additional packages;
  - unpacking packages.

Options:
  -c, --check         only check if system is ready for installation, don't execute
actions
  -h, --help          show usage
  -V, --version       show version
  -v, --verbose       verbose output
  -d, --debug         debug mode
  -y, --yes           don't ask user confirmation
  --skip-integrity    skip integrity check before unpacking

Supported variables:
  - ACCEPTED          don't ask user confirmation
  - ACTION_MODE       action from list: install, update, remove, unpack, download,
check
  - VERBOSE           verbose output
  - DEBUG             debug mode
  - SETUP_ROOT_DIR    setup root directory, default: /opt
  - SKIP_INTEGRITY    skip integrity check before unpacking
```

```
Required resources (for installation process only):
  CPU, cores: 1
  RAM, Gb:    2
  DISK, Gb:   20 (in /opt/r-vision)
  INODES:     5000 (in /opt/r-vision)
```

Структура каталогов

Установщик содержит как общие разделяемые компоненты, так и специфичные для продукта.

По умолчанию распаковка выполняется в каталог `/opt`, например `/opt/r-vision`. Корневой каталог (`/opt` по умолчанию) можно изменить с помощью переменной `SETUP_ROOT_DIR`.

Таким образом:

- `/opt` — корневой каталог установщика;
- `/opt/r-vision` — каталог производителя.

Если пользователь не имеет права записи в каталог `/opt`, необходимо предварительно создать каталог производителя и назначить его владельцем пользователя, от имени которого будет выполняться установка.

Пример 4. Пример создания каталога установки

```
sudo mkdir -pv /opt/r-vision
sudo chown -R <user>:<group> /opt/r-vision
```

Здесь:

- `<user>` — имя пользователя;
- `<group>` — группа, в которую входит пользователь.

▼ *Пример возможной структуры каталогов после распаковки*

```
/opt/r-vision/              # каталог производителя
 ├──── cms                     # каталог модуля cms
 │     ├──── helm               # helm-чарты
 │     ├──── images             # образы для helm-чартов
 │     └──── module.yaml        # манифест модуля
 ├──── common                  # общие компоненты установщика
 │     ├──── ansible            # роли и плейбуки Ansible
 │     ├──── bin                # утилиты
 │     │     ├──── evoctl
 │     │     ├──── helm
 │     │     ├──── jq
 │     │     └──── yq
 │     ├──── lib                # библиотека bash-функций
 │     ├──── logs               # логи установщика
 │     └──── platform.yaml      # манифест дистрибутива
 ├──── core                    # каталог модуля core
```

```
│   ├── helm
│   ├── images
│   └── module.yaml
├── infra                    # каталог модуля infra
│   ├── helm
│   ├── images
│   ├── menu.sh
│   ├── module.yaml
│   ├── postinstall.sh
│   └── preinstall.sh
├── playbooks                # каталог модуля playbooks
│   ├── helm
│   ├── images
│   ├── module.yaml
│   └── preinstall.sh
├── siem                     # каталог модуля siem
│   ├── helm
│   ├── images
│   ├── menu.sh
│   ├── module.yaml
│   ├── postinstall.sh
│   └── preinstall.sh
├── space                    # каталог модуля space
│   ├── helm
│   ├── images
│   ├── module.yaml
│   ├── postinstall-space.sh
│   └── preinstall-space.sh
└── vm                       # каталог модуля vm
    ├── helm
    ├── images
    └── module.yaml
```

Логирование

Все действия логируются. При каждом запуске установщика создается новый лог-файл в каталоге `logs` системы. В лог сохраняется более подробная информация, чем выводится на экран, даже в verbose-режиме. Также в лог сохраняется информация об операционной системе, ресурсах, программном окружении и переменных. Значения переменных с именами, содержащими `pass` или `private`, маскируются.

▼ *Пример системной информации в логе*

```
System information:
  OS ID:          redos
  OS name:        RED OS MUROM (7.3.4)
  Kernel:         6.1.52-1.el7.3.x86_64
  User:           user
  CPU, cores:     8
  RAM, GB:        16
App directory info:
```

```
  Directory:     /opt/r-vision
  Free space:    53G
  Free inodes:   50903398


Suitable tools:
  bash|4.0.0:          5.2.9
  gunzip:              1.9.0
  sed:                 4.5.0
  tar:                 1.30.0
  diff:                3.6.0
  ansible|2.9:         2.12.5
  kubectl:             1.26.3
  python3|3.6.8:       3.8.18
  curl:                7.85.0
  whiptail:            0.52.21
  dialog:              <not installed>

Environment:
------------
HOME=/home/user
LANG=en_US.UTF-8
LANGUAGE=en_US.UTF-8
TERM=xterm-256color
TEST_PASSWORD=<masked>
private_key=<masked>
USER=user
------------
```

Интерфейс установщика

Для взаимодействия с пользователем используется консольная диалоговая утилита `whiptail`. Если она не установлена, используется режим консоли.

У пользователя интерактивно запрашиваются необходимые параметры установки, выполняется проверка введенных значений.

Также поддерживается неинтерактивный режим установки с получением параметров из переменных окружения, определенных в env-файле.

▼ *Встроенная справка установщика*

```
Usage: evo.platform_<version>.run [OPTIONS]

Self-extracting installer for R-Vision EVO Platform, version: 1.0.0.

Available main actions:
  - new installation;
  - upgrading existing installation;
  - removing product;
  - downloading additional packages;
```

```
  - unpacking packages.

Options:
  -c, --check         only check if system is ready for installation, don't execute
actions
  -h, --help          show usage
  -V, --version       show version
  -v, --verbose       verbose output
  -d, --debug         debug mode
  -y, --yes           don't ask user confirmation
  --skip-integrity    skip integrity check before unpacking

Supported variables:
  - ACCEPTED          don't ask user confirmation
  - ACTION_MODE       action from list: install, update, remove, unpack, download,
check
  - VERBOSE           verbose output
  - DEBUG             debug mode
  - SETUP_ROOT_DIR    setup root directory, default: /opt
  - SKIP_INTEGRITY    skip integrity check before unpacking

Required resources (for installation process only):
  CPU, cores: 1
  RAM, Gb:    2
  DISK, Gb:   20 (in /opt/r-vision)
  INODES:     5000 (in /opt/r-vision)
```

Порядок работы установщика

Процесс работы установщика состоит из следующих этапов:

1. Подготовка (общий этап для всех продуктов R-Vision, работающих на платформе Kubernetes):
   - Проверка соответствия программным и аппаратным требованиям для работы установщика.
   - Проверка целостности файла установщика.
   - Обнаружение и проверка целостности загружаемых пакетов.
   - Выбор режима работы установщика:
     - Новая установка в существующий кластер Kubernetes.
     - Обновление предыдущей версии.
     - Удаление установленного экземпляра системы.
     - Распаковка имеющихся архивов.
     - Скачивание недостающих пакетов.
2. Действия, специфичные для выбранного режима работы установщика.

При выборе установки, обновления или удаления выполняется распаковка пакетов в каталог продукта или их удаление из каталога, а управление передается скрипту установщика системы.

Процесс установки системы

В данном разделе описаны действия по выполнению новой установки системы.

Установка состоит из следующих этапов:

1. Аудит системного окружения
2. Запуск установщика
3. Начало установки
4. Настройка Ansible
5. Выбор пространства имен Kubernetes
6. Настройка пула адресов балансировщика
7. Настройка доменного имени
8. Настройка PostgreSQL
9. Настройка ClickHouse
10. Настройка Prometheus
11. Настройка Loki
12. Настройка Consul
13. Настройка MinIO
14. Настройка Dkron
15. Настройка NATS
16. Настройка СУБД для работы с событиями ИБ
17. Завершение установки
18. Удаление файлов установщика
19. Авторизация в системе

Аудит системного окружения

Если на узлах кластера установлен брандмауэр firewalld, на время установки системы рекомендуется отключить его, поскольку он может препятствовать сетевому взаимодействию между подами.

Чтобы отключить firewalld:

1. Проверьте статус firewalld:

   ```
   systemctl is-active firewalld.service
   ```

2. Если команда выше вывела `active`, остановите firewalld:

   ```
   systemctl stop firewalld.service
   ```

3. Проверьте, что firewalld остановлен:

```
systemctl is-active firewalld.service
```

Команда должна вывести `inactive`.

4. Установите систему.
5. По завершении установки включите firewalld:

```
systemctl start firewalld
```

Запуск установщика

Чтобы запустить установщик системы:

1. После скачивания run-файла перейдите в директорию его хранения и запустите его с помощью bash:

```
bash ./evo.platform_<version>.run
```

Требуемая версия bash: 4.0 или выше.

Или сделайте его исполняемым и запустите непосредственно:

```
chmod +x ./evo.platform_<version>.run
./evo.platform_<version>.run
```

По умолчанию установка выполняется в каталог `/opt`. При необходимости вы можете изменить каталог установки с помощью переменной `SETUP_ROOT_DIR`:

```
SETUP_ROOT_DIR=<setup_dir> bash ./evo.platform_<version>.run
```

Здесь:

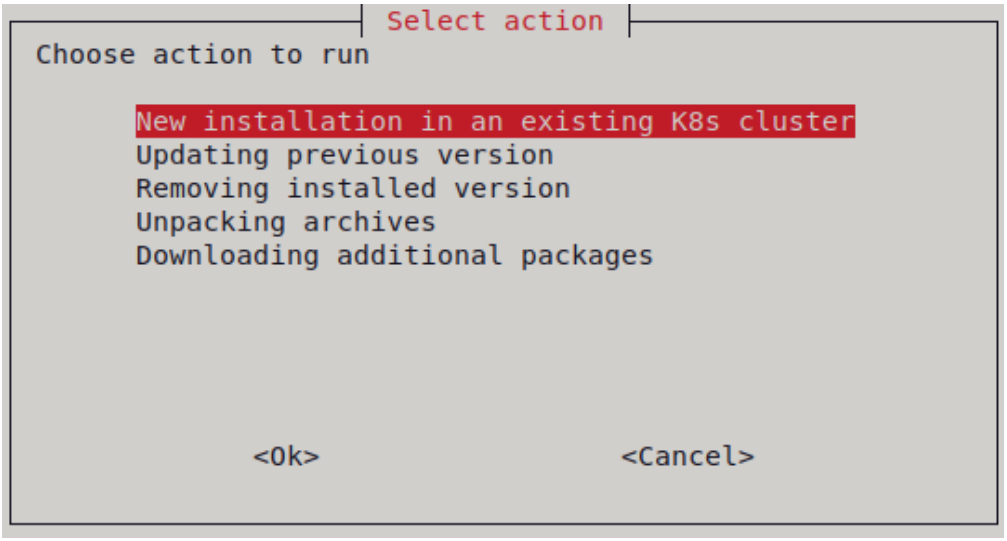
- `<setup_dir>` — директория установки.

Установщик выведет следующие сведения:

- версия системы и дата ее выпуска;
- путь к файлу хранения логов установки;
- результаты проверок:
  - удовлетворения системным требованиям;
  - наличия необходимых утилит;

- целостности дистрибутива.

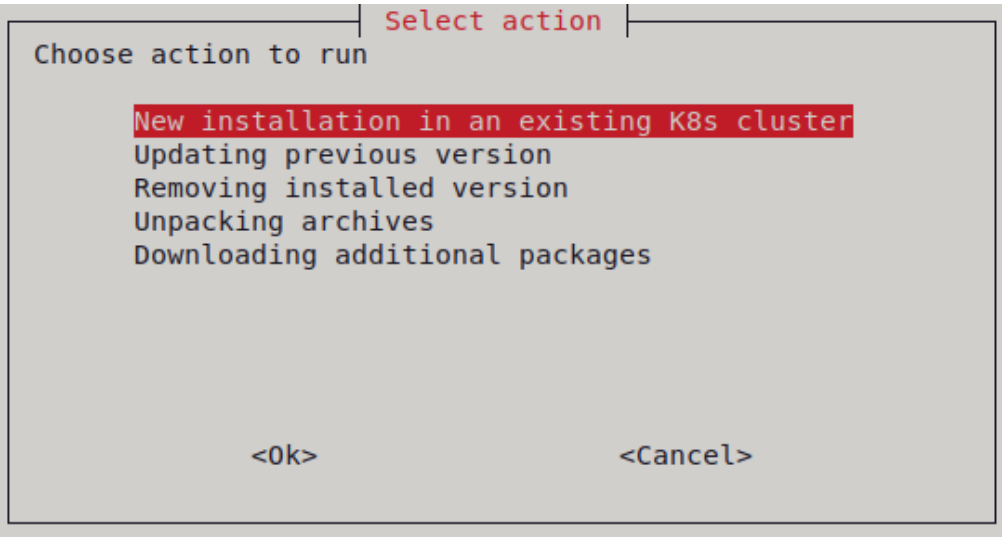
2. Выберите режим работы установщика и следуйте инструкциям на экране.



- **New installation in an existing K8s cluster** — новая установка в существующий кластер K8s.
- **Updating previous version** — обновление предыдущей версии.
- **Removing installed version** — удаление установленного экземпляра системы.
- **Unpacking archives** — распаковка имеющихся архивов без их установки.
- **Downloading additional packages** — скачивание дополнительного пакета автономного установщика Kubernetes для РЕД ОС.

Начало установки

Чтобы начать новую установку системы, запустите установщик и выберите режим работы **New installation in an existing K8s cluster**.

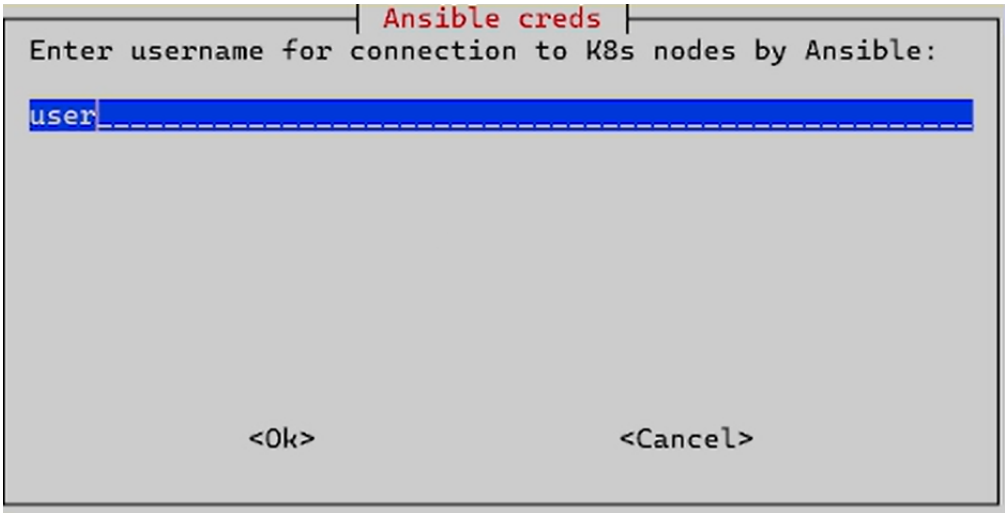


При выборе установки выполняется распаковка пакетов в каталог продукта и управление передается скрипту установщика.

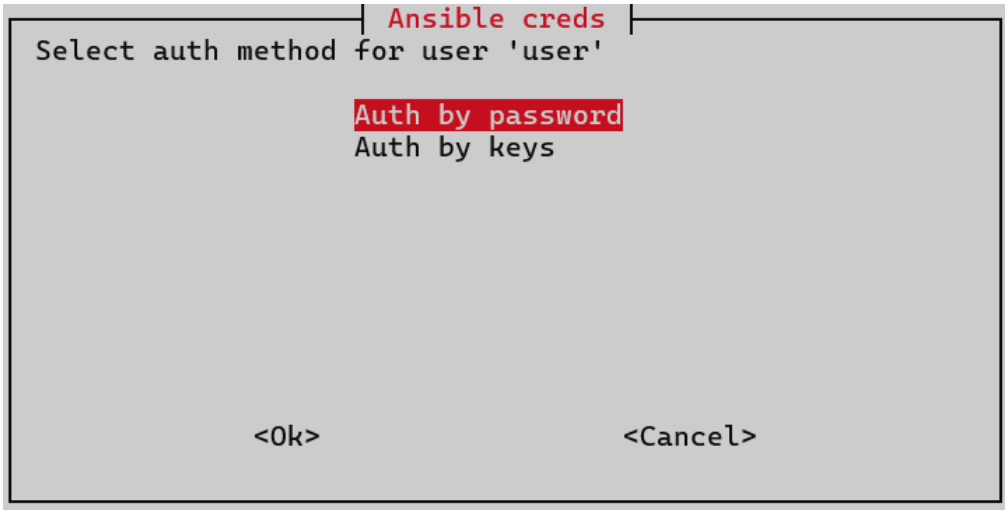
Настройка Ansible

Задайте параметры Ansible на этапе **Ansible creds**:

1. Задайте имя пользователя.



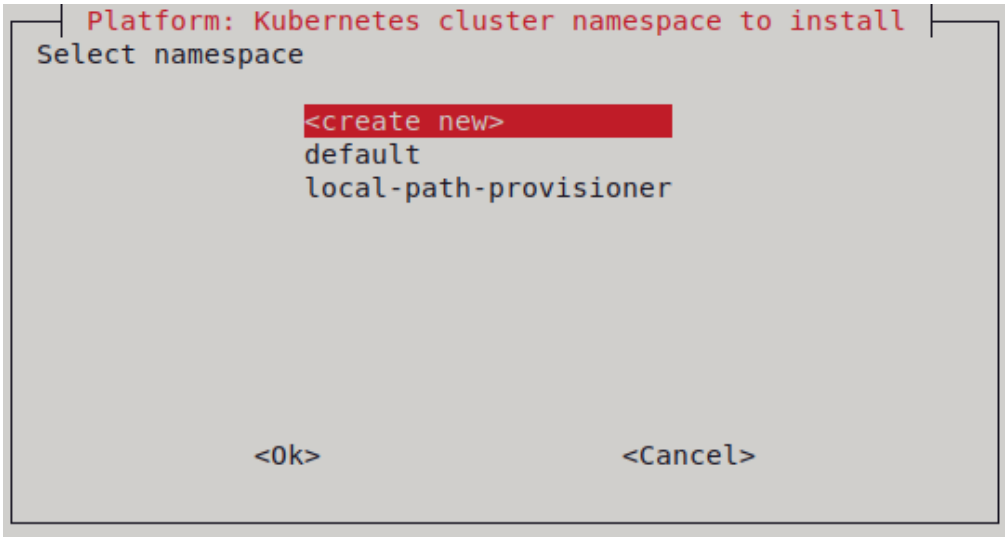
2. Выберите метод аутентификации: по паролю или через ключ.



3. Если выбрана аутентификация по паролю, задайте и подтвердите пароль пользователя.

Выбор пространства имен Kubernetes

Выберите пространство имен (namespace) Kubernetes, в которое будет установлен кластер, на этапе **Kubernetes cluster namespace to install**.

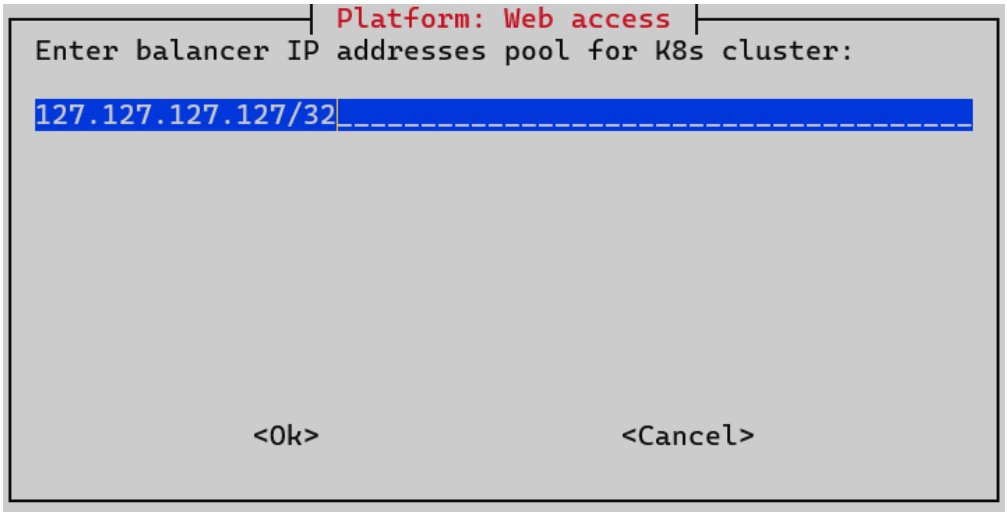


Если требуется создать новое пространство имен, выберите опцию **create new** и введите название пространства.

Настройка пула адресов балансировщика

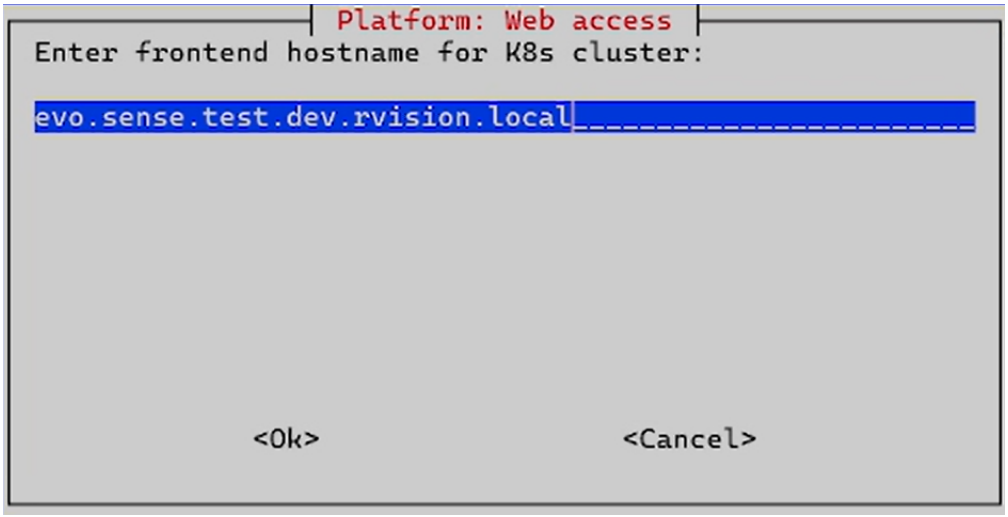
Настраивать пул адресов балансировщика требуется, только если выполняется распределенная установка в кластере, где есть хотя бы один управляющий и один рабочий узел. При установке системы на один сервер данный этап будет пропущен.

Введите пул адресов балансировщика на этапе **Web access**.



Настройка доменного имени

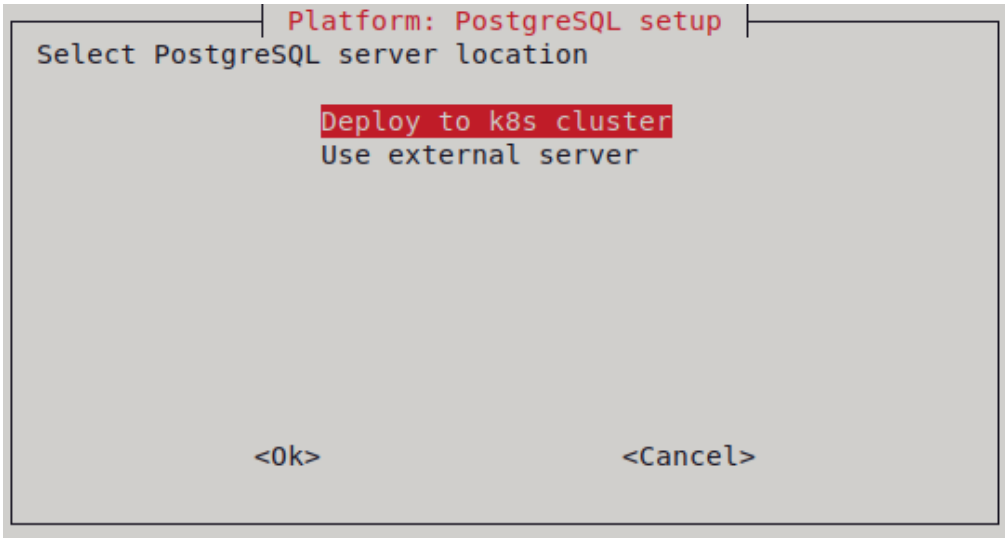
Введите доменное имя для внешних запросов на этапе **Web access**.



Настройка PostgreSQL

На данном этапе потребуется учетная запись пользователя PostgreSQL с правами на создание новых баз данных и пользователей. Для этого можно использовать суперпользователя `postgres` или создать нового.

Задайте параметры PostgreSQL на этапе **PostgreSQL setup**.



Выберите расположение:

- **Deploy to k8s cluster** — в кластере Kubernetes.
- **Use external server** — внешнее расположение на выделенном сервере.

Для установки в **кластер Kubernetes** укажите следующие параметры:

- название базы данных PostgreSQL;
- имя и пароль пользователя PostgreSQL.

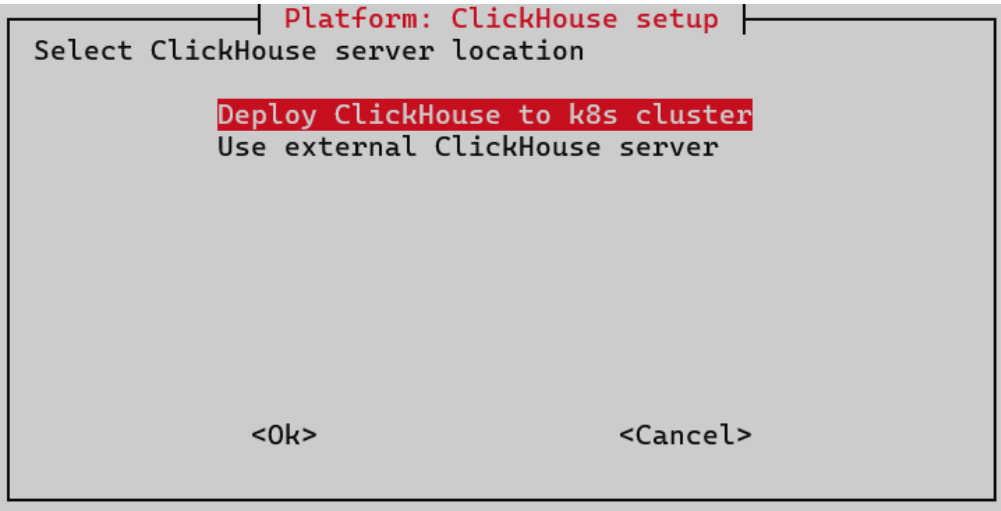
Для расположения на **выделенном сервере** укажите следующие параметры для доступа к PostgreSQL:

- имя хоста или IP-адрес сервера PostgreSQL;

- порт сервера;
- имя пользователя;
- пароль пользователя.

Настройка ClickHouse

Задайте параметры ClickHouse на этапе **ClickHouse setup**.



Выберите расположение:

- **Deploy ClickHouse to k8s cluster** — в кластере Kubernetes.
- **Use external ClickHouse server** — внешнее расположение на выделенном сервере. Для этого выделенный сервер должен быть предварительно настроен.

Для установки в **кластер Kubernetes** укажите следующие параметры:

- количество реплик и шардов ClickHouse;

> Количество реплик и шардов требуется указывать, только если в кластере установлено более одного рабочего узла.

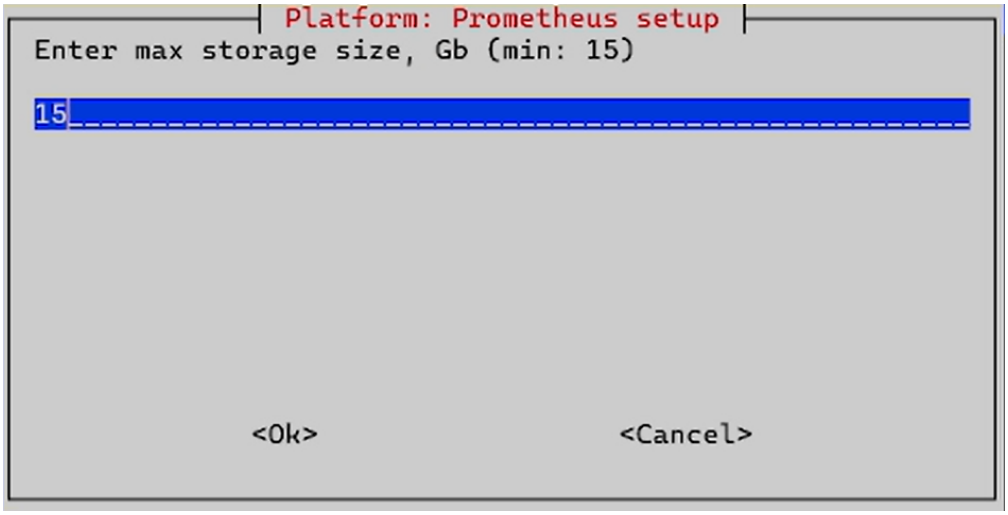
- максимальный размер хранилища ClickHouse в ГБ;
- имя пользователя ClickHouse и его пароль.

Для расположения на **выделенном сервере** задайте имя хоста, порт и gRPC-порт.

Настройка Prometheus

Prometheus — это система мониторинга, собирающая различные метрики работы кластера.

Задайте максимальный размер хранилища Prometheus на этапе **Prometheus setup**, не менее 15 ГБ.



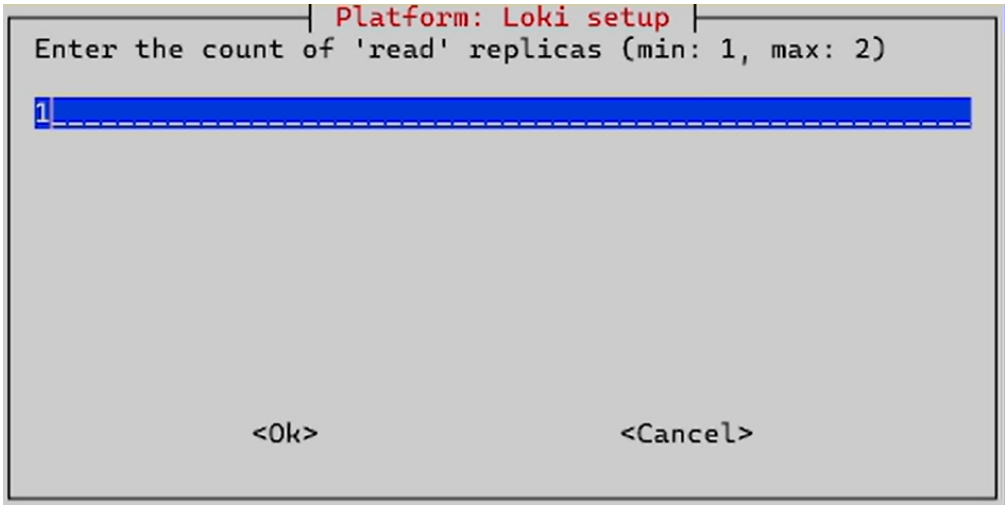
Настройка Loki

Loki — это система для хранения и просмотра логов.

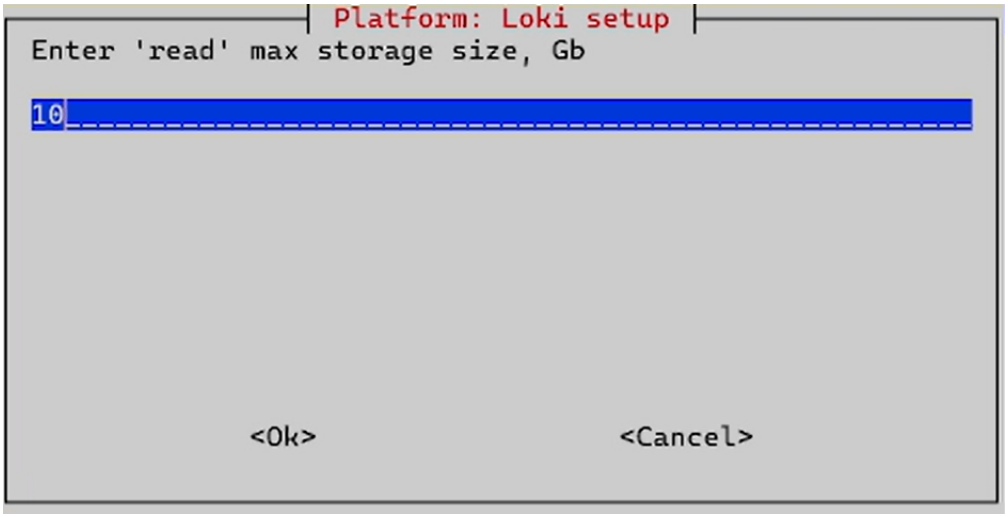
Количество реплик требуется указывать, только если в кластере установлено более одного рабочего узла.

Задайте параметры Loki на этапе **Loki setup**:

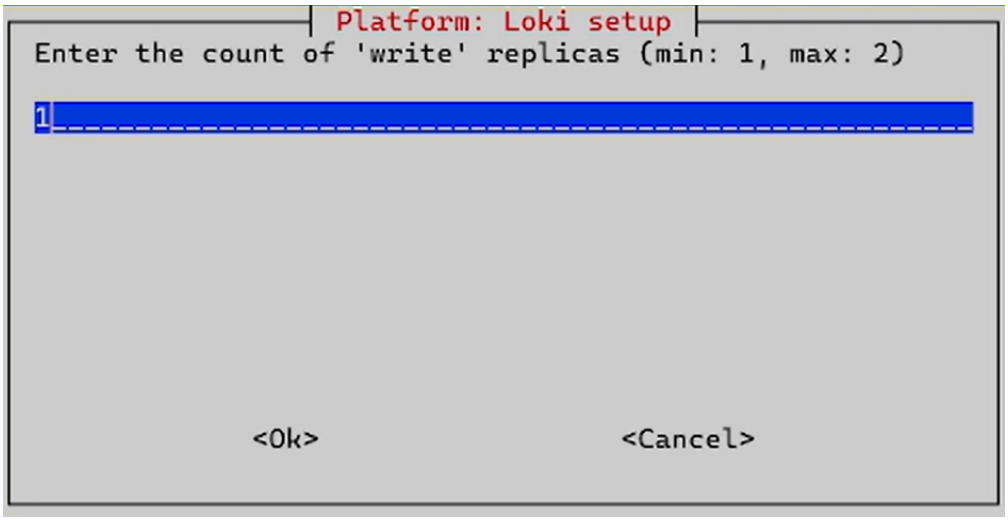
1. Задайте количество реплик хранилища для чтения логов.



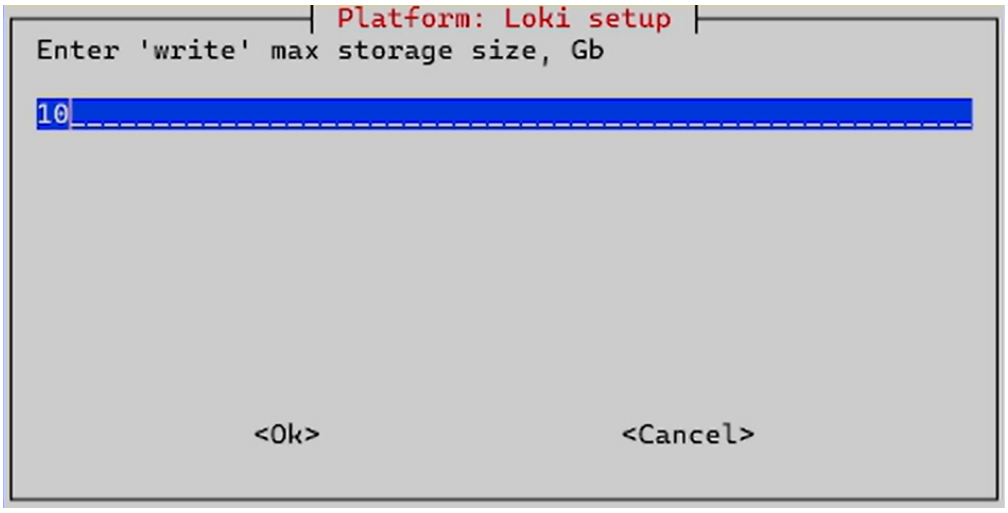
2. Задайте максимальный размер хранилища для чтения логов в ГБ.



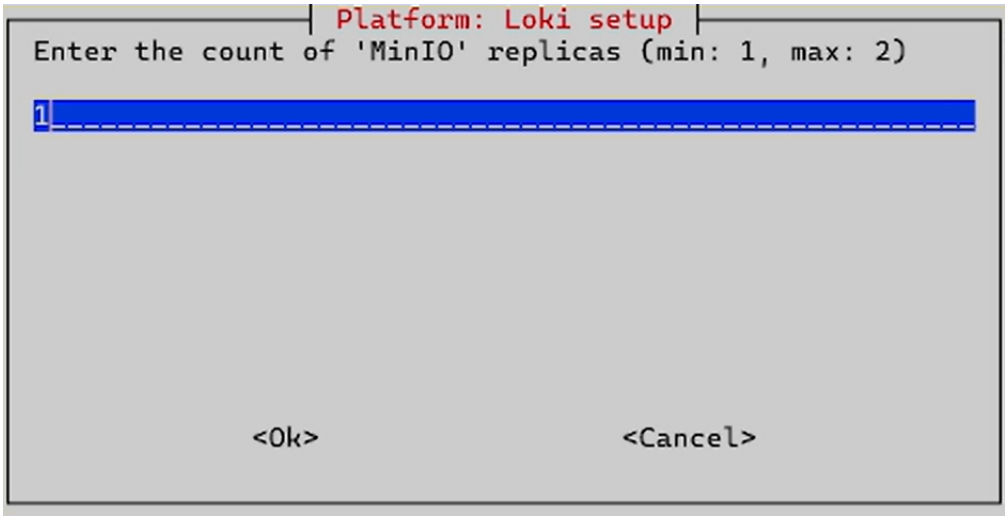
3. Задайте количество реплик хранилища для записи логов.



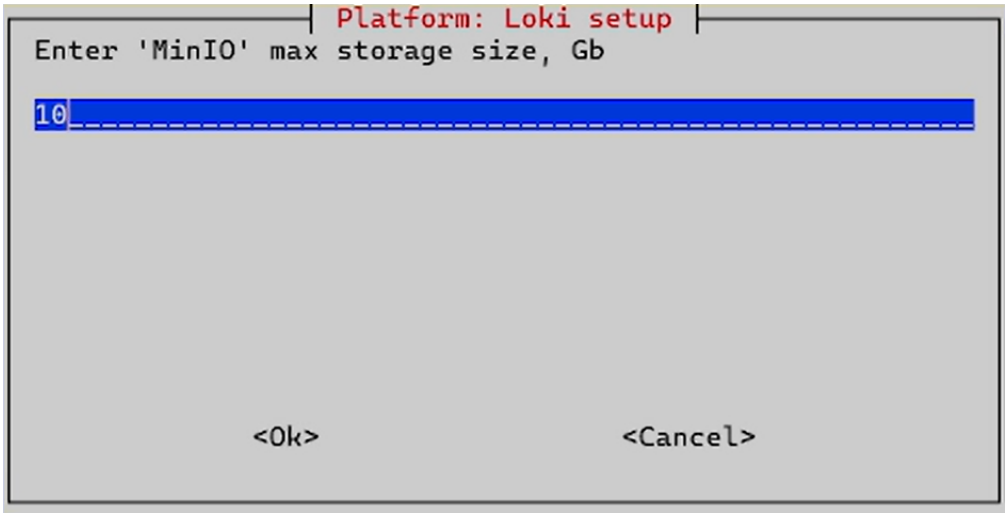
4. Задайте максимальный размер хранилища для записи логов в ГБ.



5. Задайте количество реплик MinIO.



6. Задайте максимальный размер хранилища MinIO в ГБ.

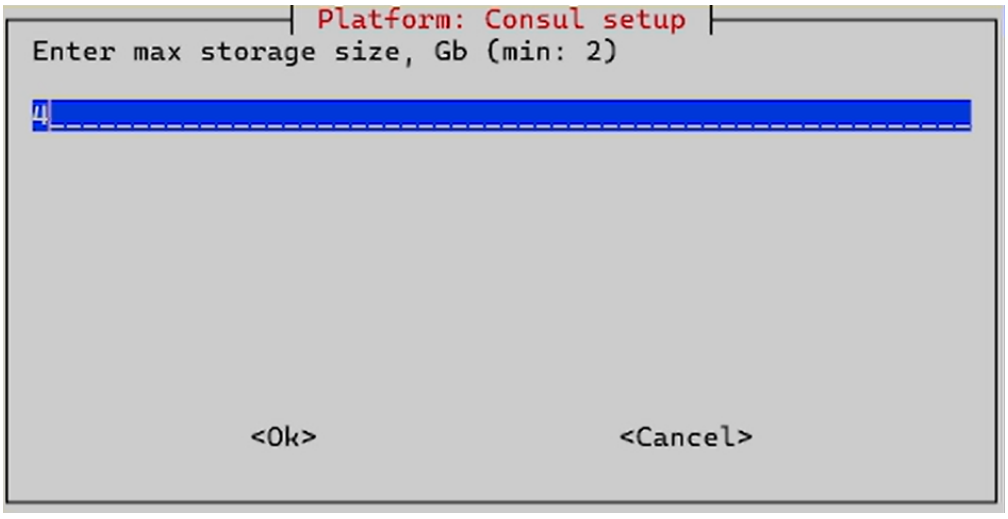


Настройка Consul

Consul — это система обнаружения сервисов (service discovery).

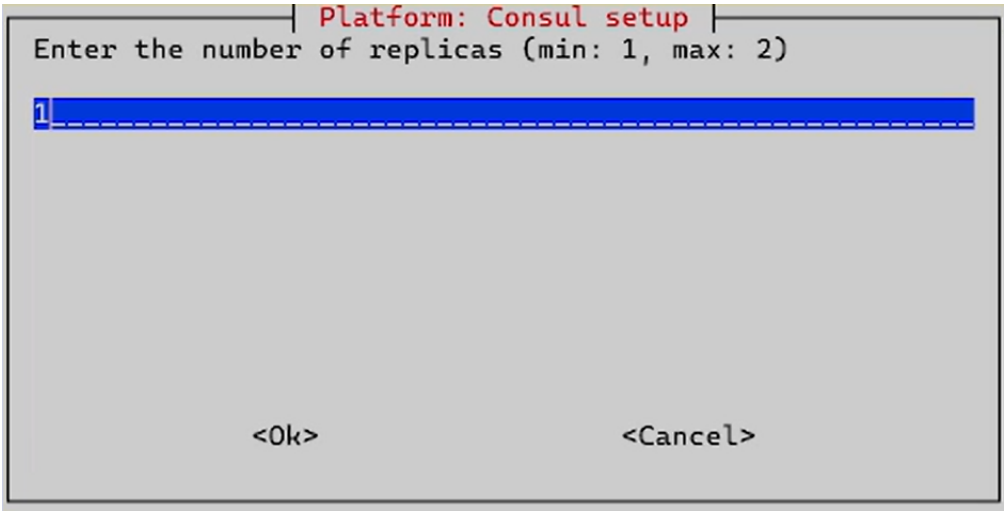
На этапе **Consul setup**:

1. Задайте максимальный размер хранилища Consul, не менее 2 ГБ.



2. Укажите количество реплик Consul.

Количество реплик требуется указывать, только если в кластере установлено более одного рабочего узла.

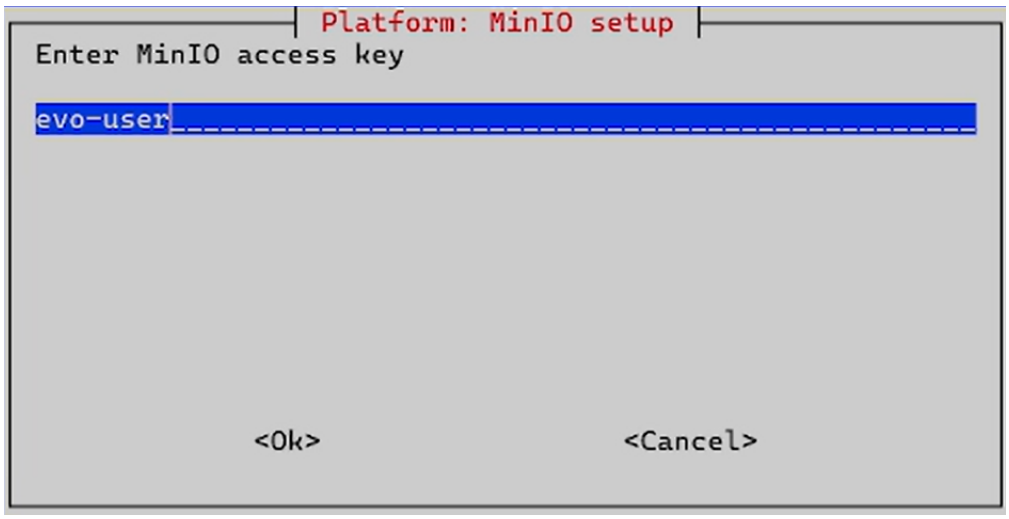


Настройка MinIO

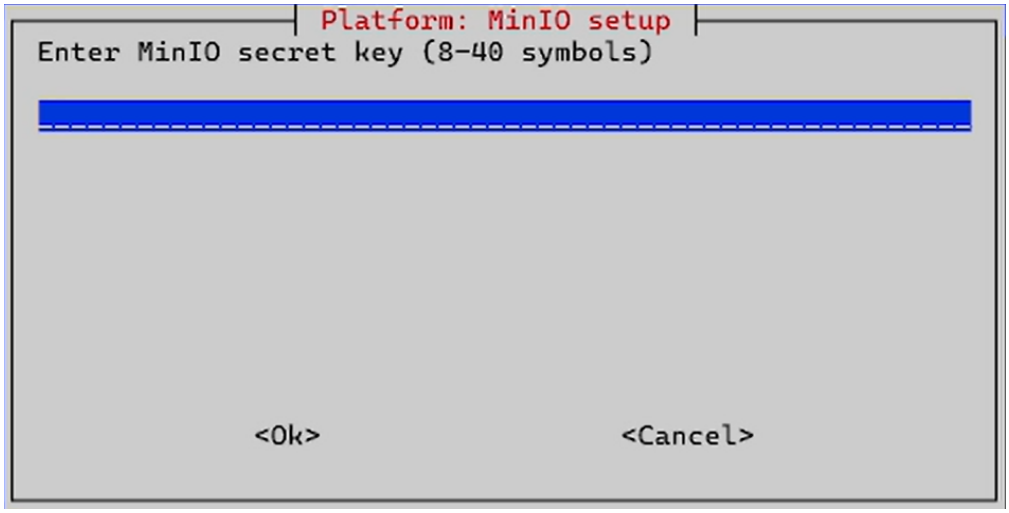
MinIO — это система объектного хранения данных, которая используется для хранения файлов конфигураций, загруженных в систему файлов, а также логов сервисов и инфраструктуры.

На этапе **MinIO setup**:

1. Задайте параметры аутентификации пользователя в MinIO.
   - **access key** — имя пользователя в MinIO.

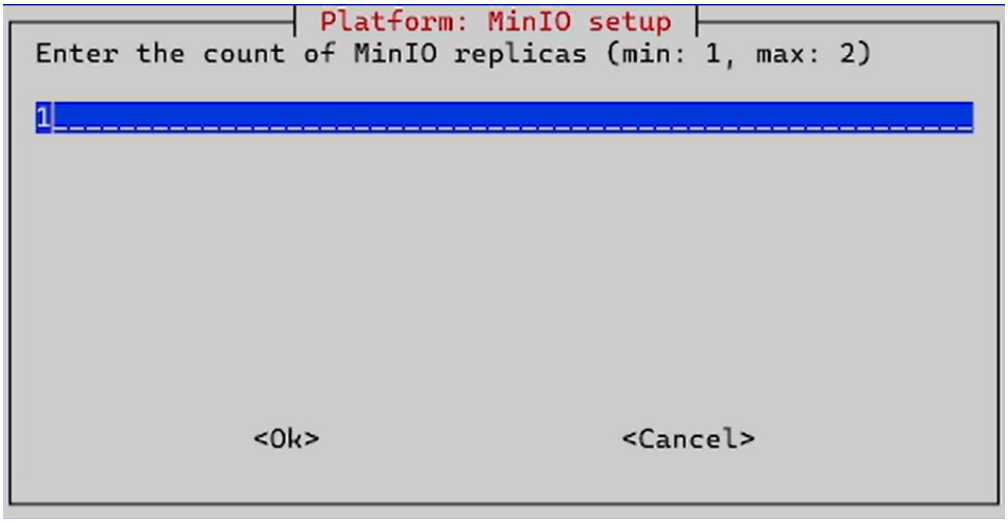


   - **secret key** — пароль пользователя в MinIO. Должен содержать от 8 до 40 знаков.



2. Укажите количество реплик MinIO.

Количество реплик требуется указывать, только если в кластере установлено более одного рабочего узла.

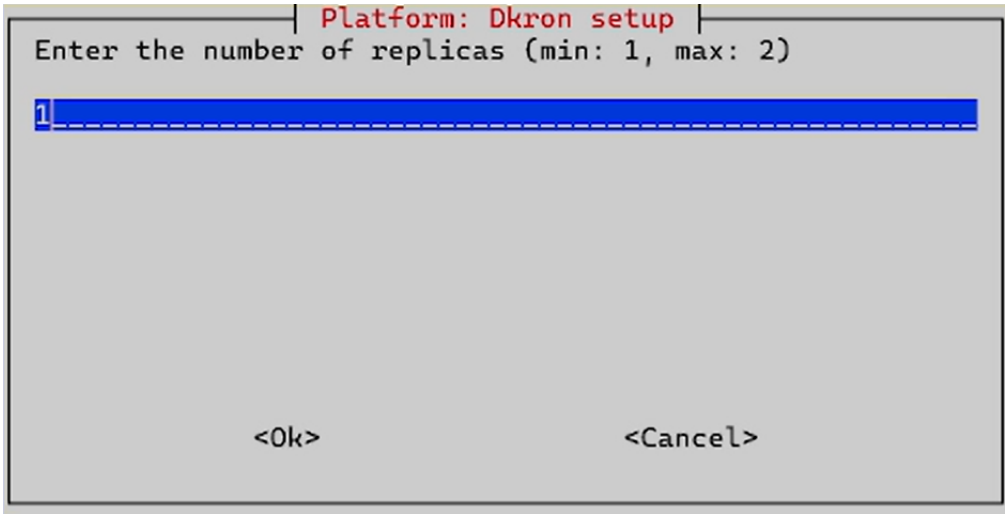


Настройка Dkron

Dkron — это распределенный планировщик задач.

Количество реплик Dkron требуется указывать, только если в кластере установлено более одного рабочего узла. В ином случае этап настройки Dkron будет пропущен.

На этапе **Dkron setup** укажите количество реплик Dkron.

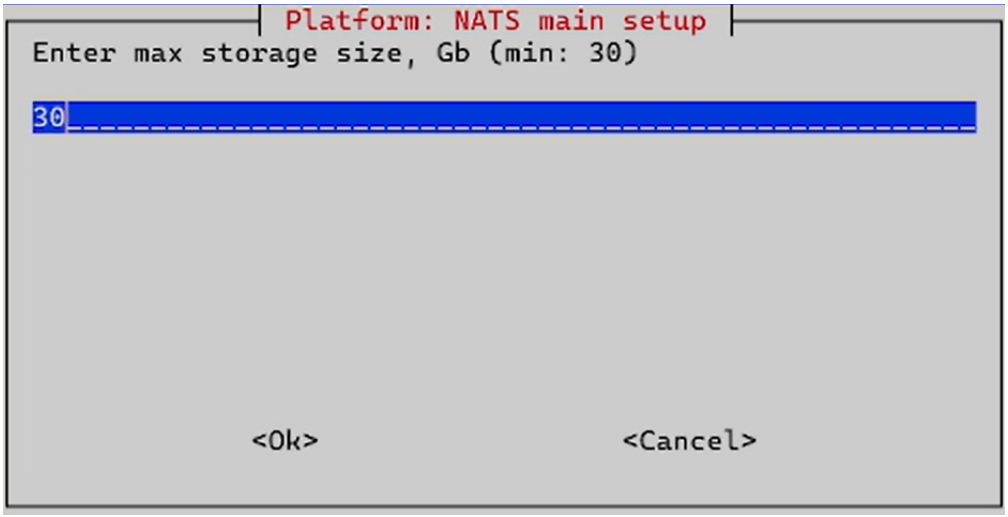


Настройка NATS

NATS — это брокер сообщений, который обеспечивает взаимодействие модулей платформы.

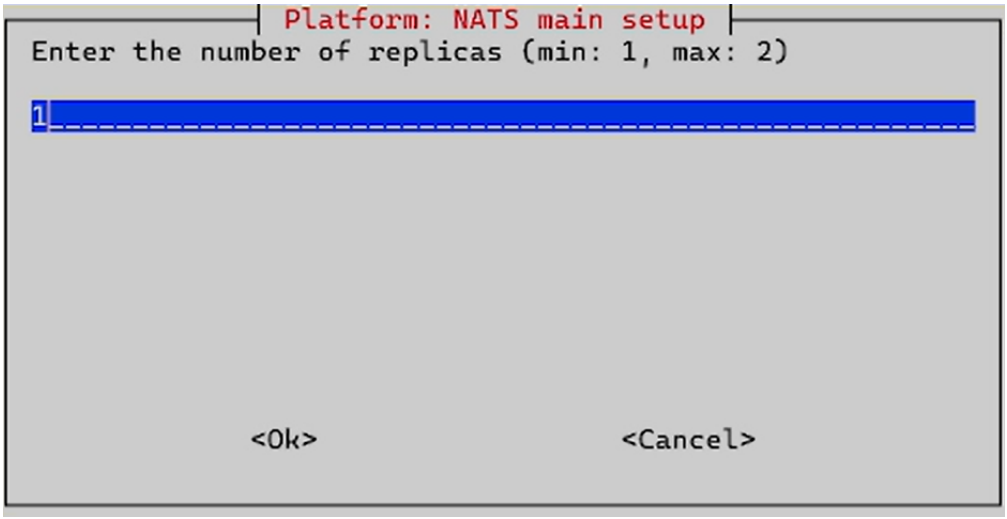
На этапе **NATS main setup** настройте основное хранилище NATS:

1. Задайте максимальный размер хранилища NATS, не менее 30 ГБ.



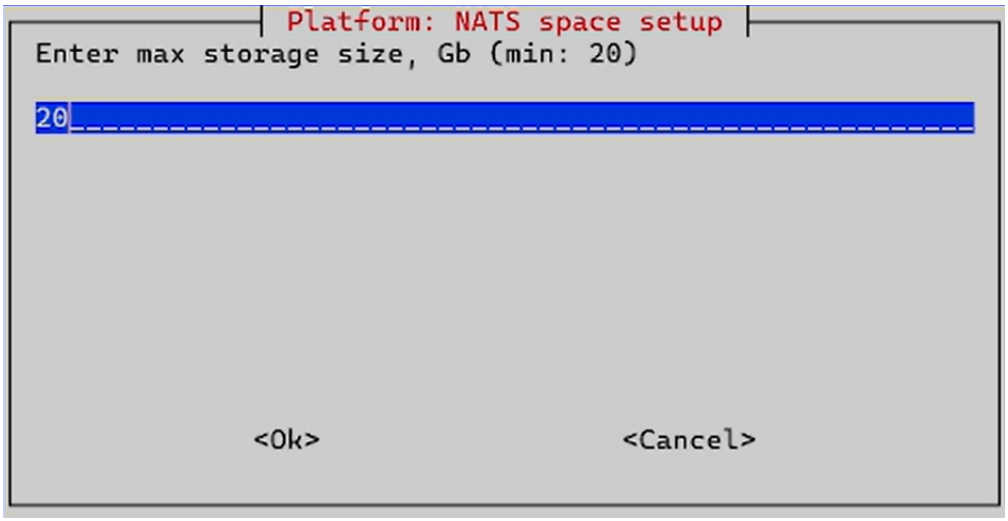
2. Укажите количество реплик NATS.

Количество реплик требуется указывать, только если в кластере установлено более одного рабочего узла.



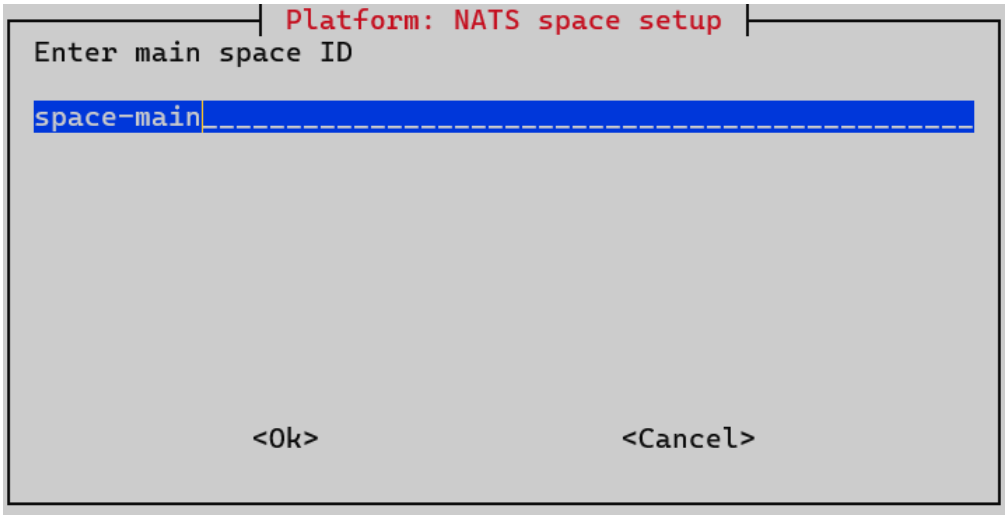
На этапе **NATS space setup** настройте хранилище NATS для управления пространствами:

1. Задайте максимальный размер хранилища NATS, не менее 20 ГБ.



2. Укажите название пространства, которое будет использовать NATS.

Допустимые символы: прописные и строчные латинские буквы, цифры, знак дефиса. Идентификатор должен содержать не более 100 символов.

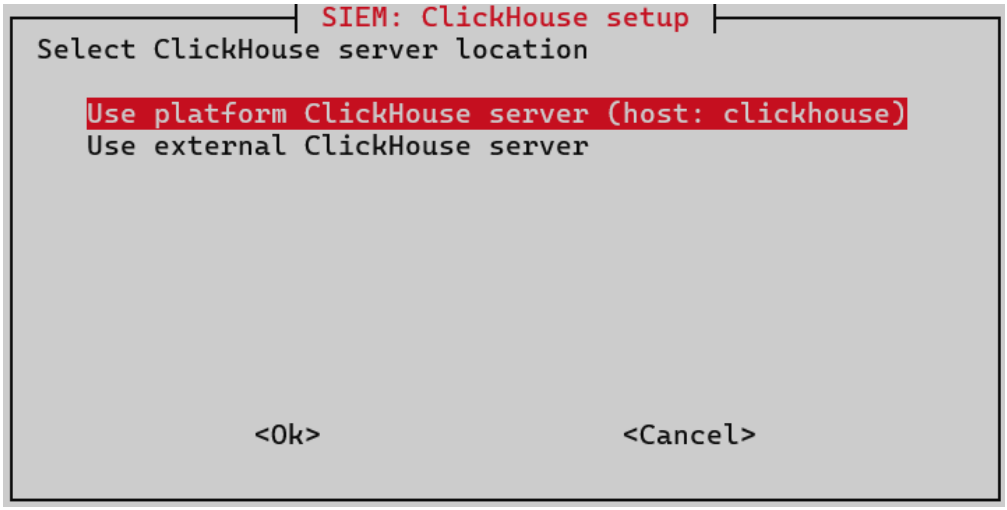


Настройка СУБД для работы с событиями ИБ

Для работы с событиями ИБ можно использовать ранее установленные инсталляции ClickHouse и PostgreSQL, общие для всех компонентов платформы, или использовать отдельные выделенные серверы.

Настройка ClickHouse для событий ИБ

На этапе **SIEM: ClickHouse setup** выберите расположение ClickHouse для событий ИБ.



Доступные варианты:

- **Use platform ClickHouse server** — использование общего сервера ClickHouse, настроенного ранее.
- **Use external ClickHouse server** — внешнее расположение на выделенном сервере.

Для работы системы необходимо, чтобы в ClickHouse были настроены три пользователя с разным набором прав:

- **owner** — пользователь с расширенным набором прав. Используется для создания, обновления и удаления таблиц (хранилищ событий), а также получения служебной информации.
- **writer** — пользователь с правами на запись. Используется в коллекторах для записи событий в ClickHouse.
- **reader** — пользователь с правами только на чтение. Используется для выполнения поисковых запросов.

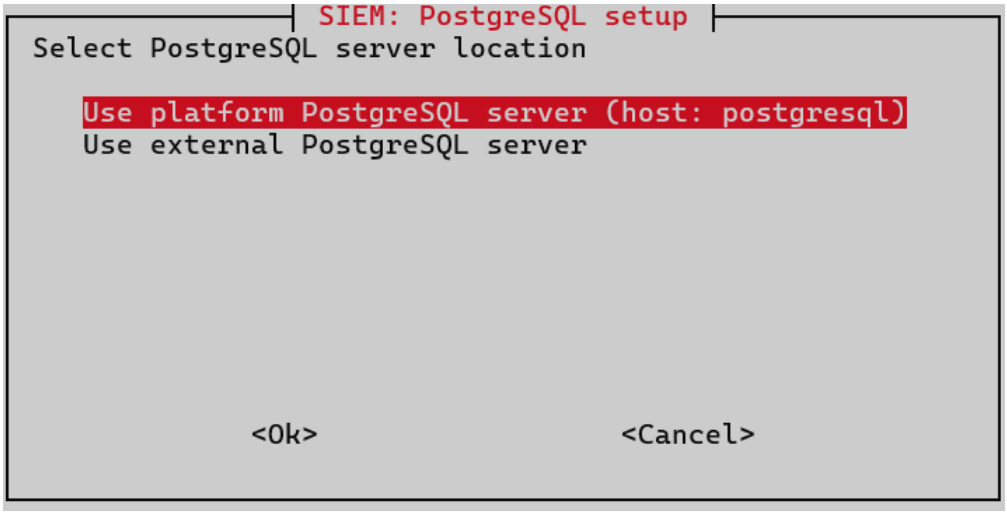
Вне зависимости от варианта размещения ClickHouse для событий ИБ следует указать пароли этих пользователей, заданные при развертывании кластера ClickHouse.

Для расположения на выделенном сервере также следует задать:

- имя хоста или IP-адрес сервера ClickHouse;
- порт сервера;
- gRPC-порт;
- имя пользователя;
- пароль пользователя.

Настройка PostgreSQL для событий ИБ

На этапе **SIEM: PostgreSQL setup** выберите расположение PostgreSQL для событий ИБ.



Доступные варианты:

- **Use platform PostgreSQL server** — использование общего сервера PostgreSQL, настроенного ранее.
- **Use external PostgreSQL server** — внешнее расположение на выделенном сервере.

Для расположения на выделенном сервере также следует задать:

- имя хоста или IP-адрес сервера PostgreSQL;
- порт сервера;
- имя базы данных;
- имя пользователя;
- пароль пользователя.

Завершение установки

Если установка прошла успешно, на экране отобразится соответствующее сообщение. Также на экран будет выведен URL-адрес для подключения к системе и учетные данные по умолчанию для входа в систему.

По завершении установки системы формируется комплект сертификатов, которые потребуются для установки сателлита.

Комплект сертификатов сохраняется в архив `$SETUP_ROOT_DIR/r-vision/nats-tls.tar.gz` (по умолчанию `/opt/r-vision/nats-tls.tar.gz`) на машине, с которой выполнялась установка.

Удаление файлов установщика

Вы можете удалить файлы и каталоги, которые необходимы только во время установки и не требуются для работы системы:

- run-файл установщика:

```
rm evo.platform_<version>.run
```

- Каталог распакованных файлов установки с подкаталогами, в том числе логи процесса установки из каталога `<installation_root>/r-vision/common/logs`:

  ```
  rm -r <installation_root>/r-vision
  ```

  Здесь:

  - `<installation_root>` — корневой каталог, в который выполнялась установка, по умолчанию `/opt`.

- Архив дополнительного пакета установщика Kubespray:

  ```
  rm -f kubespray-installer-redos.tar.gz
  ```

Авторизация в системе

Чтобы получить полное доменное имя (FQDN) системы, используйте следующую команду:

```
kubectl get secret "evo.siem.global" -n <namespace> -o
jsonpath="{.data.MODULE_DOMAIN}" | base64 -d
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.

Чтобы выполнить вход в систему:

1. В адресной строке браузера введите полное доменное имя (FQDN) системы. На экране отобразится страница авторизации.
2. Укажите логин и пароль учетной записи пользователя и нажмите на кнопку **Войти**.

По умолчанию для первого входа в систему используются учетные данные **admin**/**admin**.

Установка сателлитов

В данном разделе описан процесс установки сателлитов — компонентов системы, развернутых на дополнительных удаленных кластерах.

Кластер Kubernetes, в котором разворачиваются все системные компоненты продукта, называется **основным кластером**. Часть компонентов продукта может быть развернута на дополнительных удаленных кластерах — **сателлитах**. В отличие от основного кластера, сателлит не обладает полным спектром функциональности продукта и может работать только с теми компонентами, которые на нем запущены.

С подробной информацией о сателлитах можно ознакомиться в разделе Управление пространствами.

Основной экземпляр системы обязательно должен быть установлен перед установкой сателлита.

Чтобы установить сателлит, выполните следующие действия:

1. Убедитесь, что окружение, в котором будет установлен сателлит, соответствует техническим требованиям.
2. Разверните кластер Kubernetes, отдельный от кластера основного экземпляра системы. Процесс развертывания аналогичен развертыванию основного кластера.
3. Выполните подготовительные действия перед установкой сателлита: создайте каталог установки, настройте сетевой доступ к кластеру сателлита со стороны основного экземпляра системы и скачайте дистрибутив.
4. Установите сателлит.

Чтобы обновить сателлит, обратитесь к разделу Обновление системы. Процесс обновления сателлита аналогичен обновлению основного экземпляра системы.

Чтобы удалить сателлит, обратитесь к разделу Удаление системы. Процесс удаления сателлита аналогичен удалению основного экземпляра системы.

Технические требования к сателлитам

В данном разделе представлены минимальные технические требования для установки и функционирования сателлита системы.

- Программные требования
- Аппаратные требования

Программные требования

Поддерживаемые операционные системы

В случае сателлитов требования к операционной системе узлов кластера не предъявляются.

Требования к Kubernetes

- Минимальная версия Kubernetes: 1.24.
- Поддерживаемая среда выполнения контейнеров (CRI): containerd, CRI-O.

Выбор среды выполнения имеет значение для автономной установки без Docker-registry в инфраструктуре.

Системные утилиты

Утилита	Минимальная версия	Наличие на узлах Kubernetes и АРМ администратора
bash	4.0	требуется
gunzip	любая	требуется
sed	любая	требуется только на АРМ администратора
tar	любая	требуется только на АРМ администратора
curl	любая	требуется только на АРМ администратора
kubectl	1.24	требуется только на АРМ администратора
ansible	2.9	требуется
python	3.8	требуется

АРМ администратора — автоматизированное рабочее место администратора.

Если установка запускается непосредственно на узле кластера Kubernetes, то к нему применяются требования АРМ администратора.

Аппаратные требования

Требования для работы установщика

Свободное место на диске: от 10 ГБ.

Требования для узлов кластера Kubernetes

Возможны следующие варианты инсталляции сателлита:

- **Моно** — кластер Kubernetes из одного узла, выполняющего роль master-узла и worker-узла одновременно.
- **Распределенный** — кластер из нескольких узлов, содержащий как минимум один master-узел и один worker-узел.

Если при распределенной установке сателлита в кластере имеется только один worker-узел, то к нему применяются требования, аналогичные моно-инсталляции.

Ресурс	Моно	Распределенный, master-узел	Распределенный, worker-узел
CPU, ядер	8	4	6
RAM, ГБ	8	4	6
Место на диске, ГБ	100	100	100

Приведены требования для минимальной инсталляции сателлита, при которой возможна его нестабильная работа. Чтобы получить рекомендации по конфигурации сателлита для вашей инсталляции, обратитесь в отдел поддержки продаж по адресу Presale_team@rvision.ru.

Подготовка к установке сателлита

В данном разделе описаны подготовительные действия, которые необходимо выполнить перед установкой сателлита.

Подготовка состоит из следующих этапов:

1. Создание каталога установки.
2. Добавление TLS-сертификатов для подключения к основному экземпляру системы.
3. Настройка доступа к кластеру Kubernetes.
4. Настройка доступа к рабочим узлам кластера.
5. Настройка сетевой связности между основным экземпляром системы и сателлитом.
6. Скачивание архива с файлами для установки и проверка его целостности.

Создание каталога установки

Для работы установщика требуется распаковать содержимое дистрибутива в локальный каталог.

Если установка основного экземпляра системы и сателлита выполняется с одного АРМ администратора, то следует использовать для сателлита отдельный корневой каталог.

Например:

```
SETUP_ROOT_DIR=/opt/satellite ./evo.platform_<version>_satellite.run
```

Это позволит избежать смешивания установочных файлов основного экземпляра системы и сателлита в одном каталоге.

Далее в инструкции предполагается, что используется каталог по умолчанию.

По умолчанию распаковка выполняется в каталог `/opt`, например `/opt/r-vision`. Корневой каталог (`/opt` по умолчанию) можно изменить с помощью переменной `SETUP_ROOT_DIR`.

Таким образом:

- `/opt` — корневой каталог установщика;
- `/opt/r-vision` — каталог производителя.

Если пользователь не имеет права записи в каталог `/opt`, необходимо предварительно создать каталог производителя и назначить его владельцем пользователя, от имени которого будет выполняться установка.

Пример 5. Пример создания каталога установки

```
sudo mkdir -pv /opt/r-vision
sudo chown -R <user>:<group> /opt/r-vision
```

Здесь:

- `<user>` — имя пользователя;
- `<group>` — группа, в которую входит пользователь.

Добавление TLS-сертификатов для подключения к основному экземпляру системы

По завершении установки основного экземпляра системы создается архив TLS-сертификатов `$SETUP_ROOT_DIR/r-vision/nats-tls.tar.gz`. Этот архив необходимо скопировать в каталог установки сателлита до начала его установки:

```
cp -v <path_to_nats_tls> /opt/r-vision/
```

Здесь:

- `<path_to_nats_tls>` — путь к файлу `nats-tls.tar.gz` на машине, с которой выполнялась установка основного экземпляра системы.

Настройка доступа к кластеру Kubernetes

До начала установки должен быть развернут кластер Kubernetes.

Установка может выполняться как с одного из управляющих узлов кластера Kubernetes, так и с машины администратора кластера. У пользователя, от имени которого будет запускаться установщик, должен быть настроен доступ к кластеру Kubernetes.

Вы можете настроить доступ в конфигурационном файле утилиты kubectl — `$HOME/.kube/config` или задать путь к конфигурации в переменной `KUBECONFIG`:

```
export KUBECONFIG=<path_to_confg>
```

Здесь:

- `<path_to_config>` — путь к конфигурации kubectl.

Проверить доступ можно командой:

```
kubectl get nodes
```

Настройка доступа к рабочим узлам кластера

Для загрузки образов контейнеров на рабочие узлы установщик использует Ansible. Необходимо настроить SSH-доступ с машины установщика до рабочих узлов под единой учетной записью. Поддерживается доступ по ключам или паролю.

Учетная запись должна иметь права суперпользователя или иметь возможность повышения прав до суперпользователя.

Настройка сетевой связности между основным экземпляром системы и сателлитом

Следует разрешить TCP-подключения со стороны кластера, в котором установлен основной экземпляр системы, ко всем узлам кластера сателлита.

Допустимые значения номера порта подключения: `30072–32767`. Значение по умолчанию: `30072`. Это значение может быть изменено в процессе установки сателлита.

Получение дистрибутива

Скачайте архив с установочными файлами evo.platform_<version>_satellite.run и проверьте его контрольную сумму. Ссылка на дистрибутив предоставляется по запросу в службу поддержки по адресу support@rvision.ru.

```
# Получение дистрибутива.
curl -O
https://files.rvision.ru/platform/<random_part>/evo.platform_<version>_satellite.run

# Получение контрольной суммы.
curl -O
https://files.rvision.ru/platform/<random_part>/evo.platform_<version>_satellite.run.s
ha256

# Проверка контрольной суммы.
sha256sum -c evo.platform_<version>_satellite.run.sha256

# При успешной проверке контрольной суммы будет выведено следующее сообщение.
evo.platform_<version>_satellite.run: OK

# При нарушении контрольной суммы будет выведено следующее сообщение.
evo.platform_<version>_satellite.run: FAILED
```

При выявлении нарушения контрольной суммы выполнять установку **недопустимо**.

Необходимо убедиться, что скачивание run-файла дистрибутива было полностью завершено, и повторно проверить контрольную сумму.

При подтверждении нарушения контрольной суммы следует сообщить в техническую поддержку по адресу support@rvision.ru.

Процесс установки сателлита

В данном разделе описаны действия по выполнению новой установки сателлита.

Установка состоит из следующих этапов:

1. Запуск установщика
2. Настройка Ansible

3. Выбор пространства имен Kubernetes
4. Настройка NATS
5. Завершение установки
6. Подключение к пространству сателлита

Запуск установщика

Чтобы запустить установщик сателлита:

1. После скачивания run-файла перейдите в директорию его хранения и запустите его с помощью bash:

   ```
   bash ./evo.platform_<version>_satellite.run
   ```

   Требуемая версия bash: 4.0 или выше.

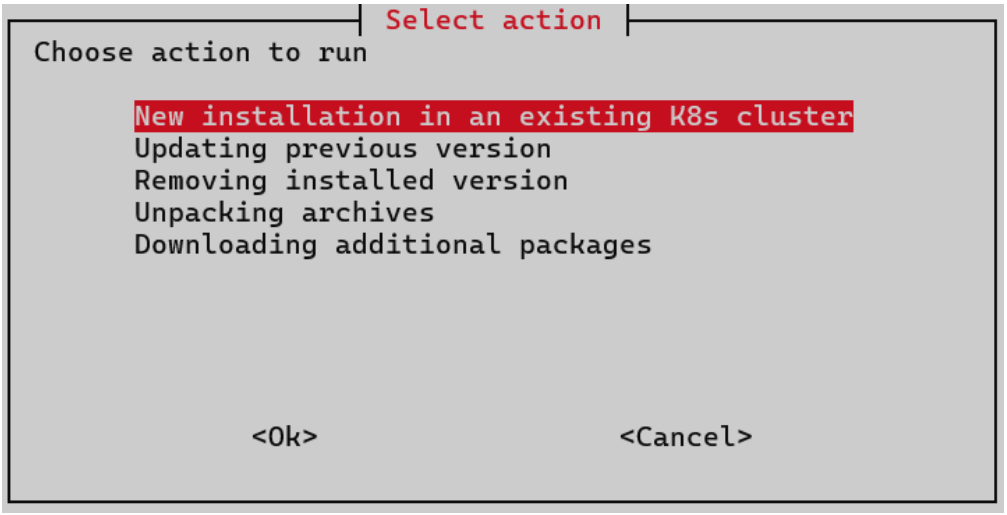
   Или сделайте его исполняемым и запустите непосредственно:

   ```
   chmod +x ./evo.platform_<version>_satellite.run
   ./evo.platform_<version>_satellite.run
   ```

   Установщик выведет следующие сведения:

   - версия системы и дата ее выпуска;
   - путь к файлу хранения логов установки;
   - результаты проверок:
     - удовлетворения системным требованиям;
     - наличия необходимых утилит;
     - целостности дистрибутива.

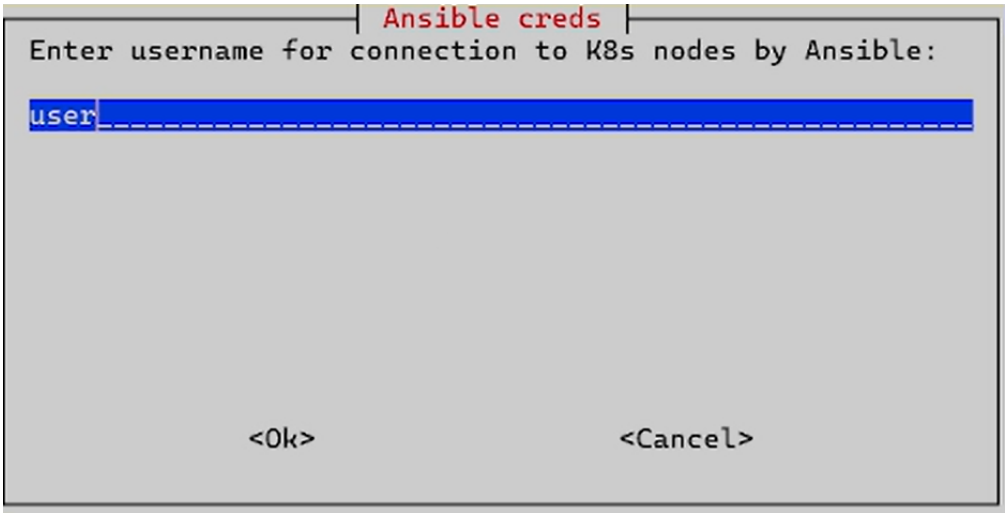
2. На этапе **Select action** выберите действие **New installation in an existing K8s cluster**:



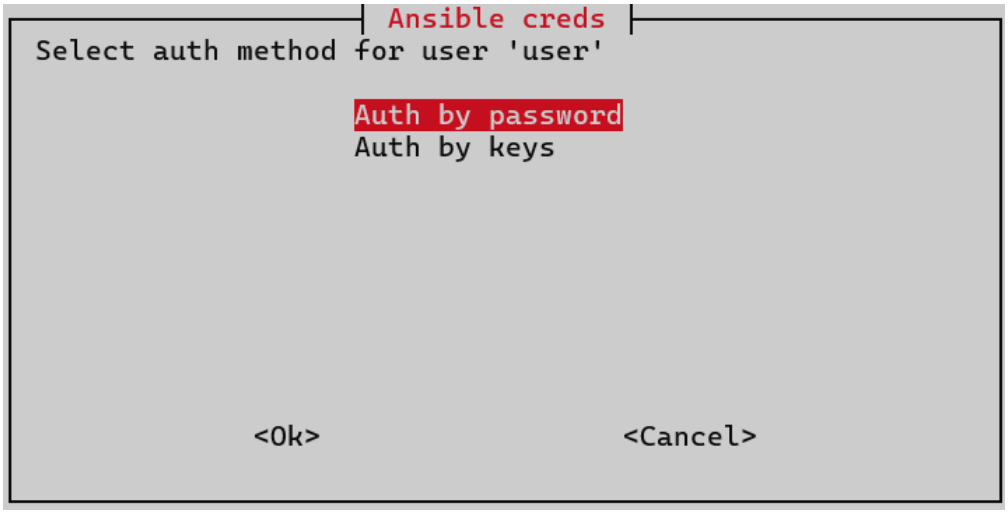
Настройка Ansible

Задайте параметры Ansible на этапе **Ansible creds**:

1. Задайте имя пользователя.



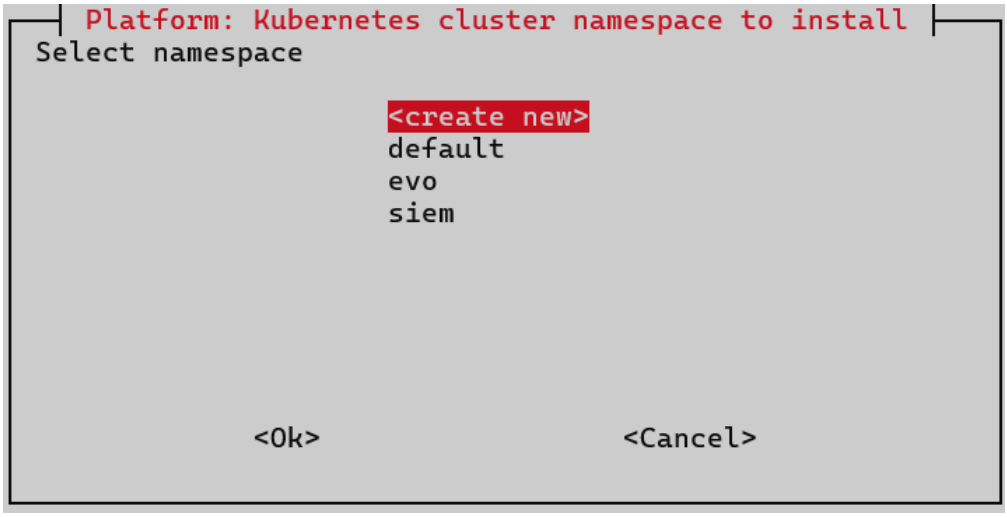
2. Выберите метод аутентификации: по паролю или через ключ.



3. Если выбрана аутентификация по паролю, задайте и подтвердите пароль пользователя.

Выбор пространства имен Kubernetes

Выберите пространство имен (namespace) Kubernetes, в которое будет установлен сателлит, на этапе **Kubernetes cluster namespace to install**.

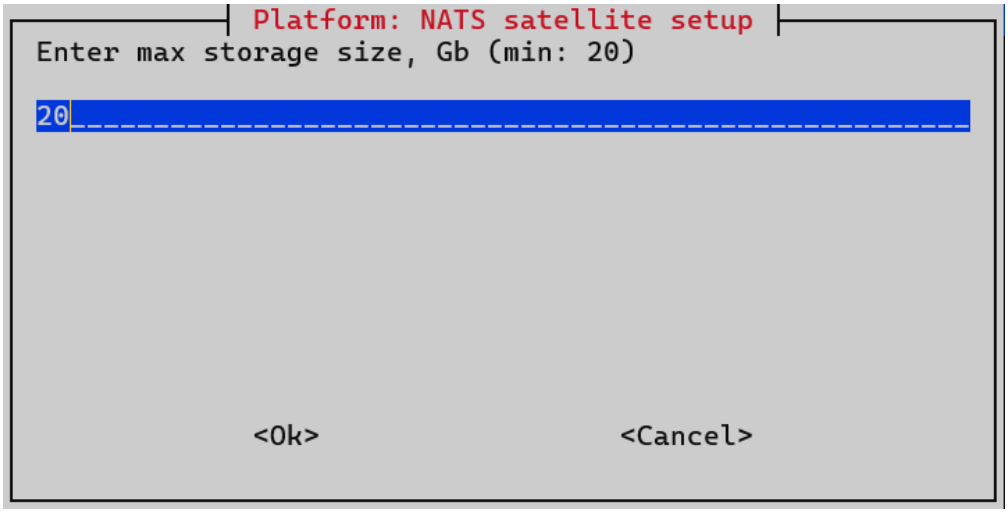


Если требуется создать новое пространство имен, выберите опцию **create new** и введите название пространства.

Настройка NATS

На этапе **NATS satellite setup**:

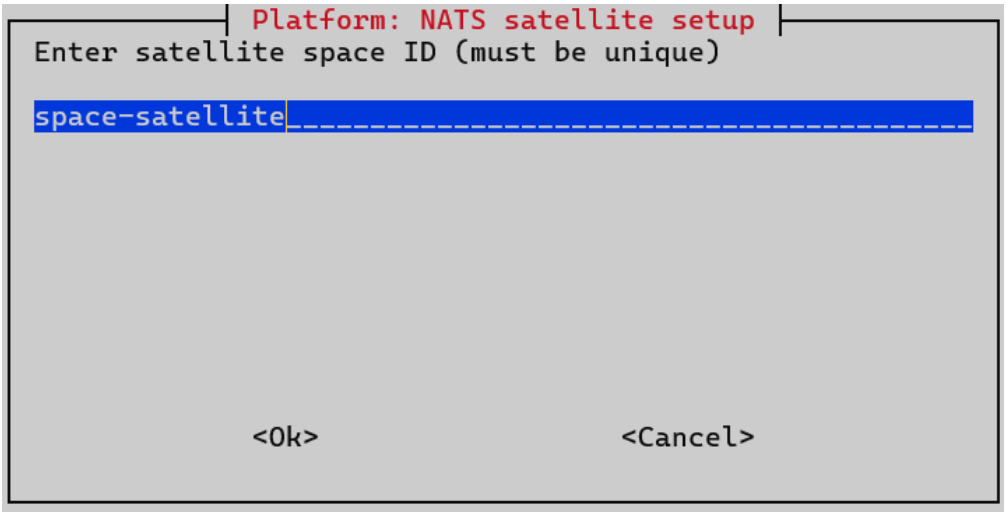
1. Укажите максимальный размер хранилища NATS, не менее 20 ГБ.



2. Укажите уникальный идентификатор пространства сателлита.

   С помощью одного дистрибутива можно установить несколько сателлитов, однако у каждого из них идентификатор пространства должен быть уникальным.

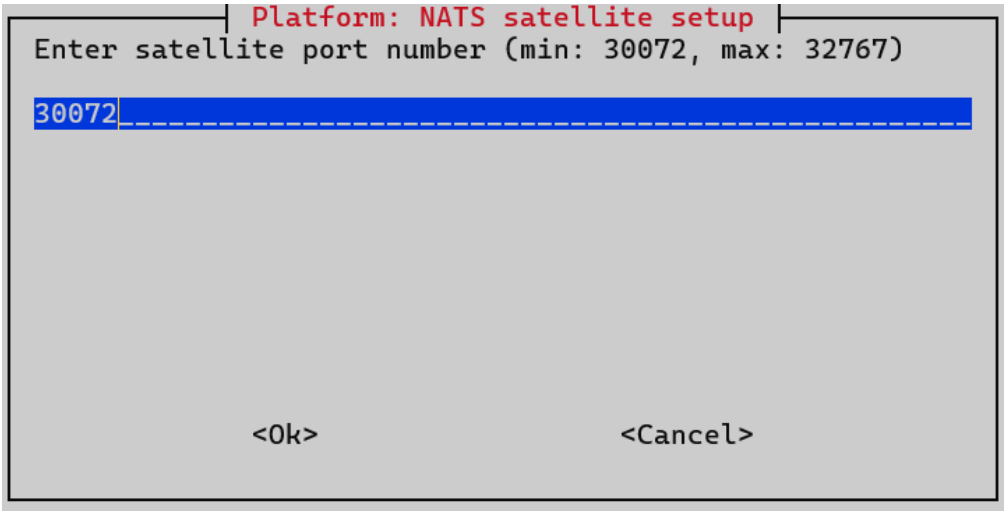
   Допустимые символы: прописные и строчные латинские буквы, цифры, знак дефиса. Идентификатор должен содержать не более 100 символов.



Сохраните идентификатор пространства, он понадобится для подключения к пространству сателлита из интерфейса системы.

3. Укажите номер TCP-порта, по которому основной экземпляр системы будет подключаться к сателлиту. Данный порт будет открыт на всех узлах кластера сателлита.

Допустимые значения номера порта подключения: `30072–32767`. Значение по умолчанию: `30072`. Сателлиты могут использовать одинаковые порты при условии установки в разные кластеры Kubernetes.



Сохраните номер порта, он понадобится для подключения к пространству сателлита из интерфейса системы.

Завершение установки

По завершении установки в терминале отобразится сообщение, содержащее идентификатор пространства и номер порта установленного сателлита, например:

```
To add this Satellite as a new space of main instance use values:
  space id: space-satellite
  space port: 30072
```

Подключение к пространству сателлита

Управление сателлитом осуществляется за счет запускаемых на нем сервисов через веб-интерфейс основного экземпляра системы. Чтобы подключиться к сателлиту, необходимо создать подключение к его пространству. Для этого:

1. Войдите в веб-интерфейс основного экземпляра системы.
2. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**.
3. Нажмите на кнопку **Создать** (┼). Отобразится окно создания подключения.
4. Введите идентификатор пространства сателлита в поле **ID**.
5. Введите название подключения. Название может быть любым.
6. В поле **IP или домен** введите IP-адрес или доменное имя (FQDN) одного из узлов кластера, в котором развернут сателлит.
7. В поле **Порт** введите номер TCP-порта, указанного на этапе Настройка NATS установки сателлита.
8. При необходимости введите описание подключения к пространству.
9. Нажмите на кнопку **Создать**. Система создаст подключение к пространству и отобразит соответствующее уведомление. Подключение отобразится в списке раздела **Настройки** → **Пространства и сервисы** → **Пространства**.

У созданного подключения отобразится статус **Подключается**. По завершении подключения статус изменится на **Активен**.

Время на инициализацию подключения зависит загруженности кластеров и каналов связи и может достигать 5—10 минут.

Обновление системы

В данном разделе описаны действия по обновлению системы или сателлита.

Обновление состоит из следующих этапов:

1. Подготовка к обновлению
2. Обновление системы
3. Авторизация в системе

Процесс обновления сателлита аналогичен обновлению основного экземпляра системы, однако после обновления сателлита может потребоваться перезапуск выполняющихся на нем коллекторов и сервисов из интерфейса основного экземпляра системы.

Подготовка к обновлению

Перед обновлением системы до новой версии выполните подготовительные действия:

1. Проверьте соответствие программных и аппаратных требований для работы установщика новой

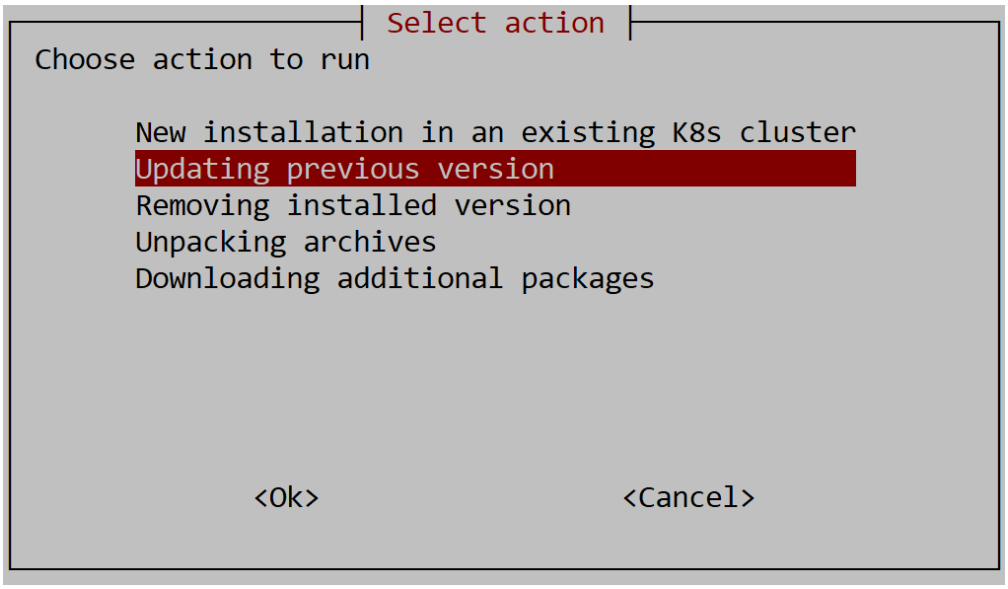
версии системы.

2. Скачайте архив с файлами для установки новой версии системы и проверьте его целостность.

Обновление системы

Чтобы обновить систему до новой версии:

1. Запустите установщик и выберите режим установки **Updating previous version**.



   При выборе обновления выполняется распаковка пакетов в каталог продукта и управление передается скрипту установщика.

2. Выберите пространство имен, в котором установлена система.
3. Задайте параметры Ansible:
   a. Имя пользователя.
   b. Метод аутентификации: через ключ или по паролю.
   c. Пароль: задайте и подтвердите пароль пользователя.

После обновления системы убедитесь, что у вас установлена поддерживаемая версия ClickHouse. Если ваша версия ClickHouse отличается от поддерживаемой, обновите ClickHouse до требуемой версии.

После обновления на экран будет выведен URL-адрес для подключения к системе.

Авторизация в системе

Чтобы выполнить вход в систему:

1. В адресной строке браузера введите URL-адрес, выведенный на экран после обновления системы. На экране отобразится страница авторизации.
2. Укажите логин и пароль учетной записи пользователя и нажмите на кнопку **Войти**.

Обновление ClickHouse

Новые версии ClickHouse не устанавливаются автоматически, обновление требуется выполнять вручную.

Поддерживаемые версии ClickHouse: 24.7 и выше.

Рекомендуется устанавливать ClickHouse максимальной поддерживаемой версии. Более новые версии ClickHouse обладают большим объемом функциональности и отличаются более высокой производительностью.

Для корректной работы системы необходимо использовать ClickHouse поддерживаемой версии. Установка неподдерживаемых версий может привести к непредвиденным последствиям.

Способ обновления зависит от расположения ClickHouse:

- Обновление ClickHouse в кластере Kubernetes
- Обновление ClickHouse на выделенном сервере

Обновление ClickHouse в кластере Kubernetes

Здесь и далее `<namespace>` — пространство имен, в котором установлен кластер.

Чтобы обновить ClickHouse в кластере Kubernetes:

1. Добавьте к переменной окружения `PATH` путь до директории `/opt/r-vision/common/bin`, в которой располагаются утилиты helm, kubectl и jq:

   ```
   export PATH="$PATH:/opt/r-vision/common/bin"
   ```

2. Перейдите в директорию установки ClickHouse:

   ```
   cd /opt/r-vision/common/helm/stateful/clickhouse
   ```

3. Сохраните параметры текущей версии ClickHouse:

   ```
   ADMIN_PASSWORD=$(kubectl get secret -n <namespace> clickhouse -o
   jsonpath="{.data.admin-password}" | base64 -d)
   SYNC_PASSWORD=$(helm get values -n <namespace> clickhouse -o json | jq -r
   '.clusterSettings.password')
   PVC_SIZE=$(helm get values -n <namespace> clickhouse -o json | jq -r
   '.persistence.size')
   REPLICAS=$(helm get values -n <namespace> clickhouse -o json | jq -r
   '.replicaCount')
   SHARDS=$(helm get values -n <namespace> clickhouse -o json | jq -r '.shards')
   ```

4. Проверьте корректность команды обновления с помощью опции `--dry-run`:

```
helm upgrade -n <namespace> clickhouse ./ -f ./custom_values.yaml --set
auth.password="${ADMIN_PASSWORD:?}" --set shards="${SHARDS:?}" --set
replicaCount="${REPLICAS:?}" --set persistence.size="${PVC_SIZE:?}" --set
clusterSettings.password="${SYNC_PASSWORD:?}" --dry-run
```

Данная команда только проверяет корректность команды обновления, но не выполняет само обновление.

5. Если при выполнении шага 4 не возникло ошибок, запустите обновление ClickHouse:

```
helm upgrade -n <namespace> clickhouse ./ -f ./custom_values.yaml --set
auth.password="${ADMIN_PASSWORD:?}" --set shards="${SHARDS:?}" --set
replicaCount="${REPLICAS:?}" --set persistence.size="${PVC_SIZE:?}" --set
clusterSettings.password="${SYNC_PASSWORD:?}"
```

6. Убедитесь, что обновление выполнилось успешно:

```
kubectl get pod -n <namespace> -l app.kubernetes.io/name=clickhouse --watch
```

После выполнения команды состояние каждого пода ClickHouse начнет выводиться в консоль в реальном времени. При успешном обновлении все поды будут иметь статус `Running`.

Чтобы прекратить наблюдение за состоянием подов, нажмите сочетание клавиш CTRL+C.

Обновление ClickHouse на выделенном сервере

Действия выполняются на каждом узле кластера ClickHouse.

Чтобы обновить ClickHouse на выделенном сервере:

1. Скачайте RPM-пакеты clickhouse-client, clickhouse-common-static и clickhouse-server нужной версии из официального репозитория ClickHouse.
2. Запустите обновление:

```
dnf install clickhouse-client-x.y.z.rpm clickhouse-common-static-x.y.z.rpm
clickhouse-server-x.y.z.rpm
```

Здесь `x.y.z` — версия скачанных пакетов.

3. Перезапустите сервис ClickHouse:

```
systemctl restart clickhouse-server
```

После обновления ClickHouse необходимо обновить права для пользователей owner, writer и reader.

Обновление прав пользователей

Для работы системы необходимо, чтобы в ClickHouse были настроены три пользователя с разным набором прав:

- **owner** — пользователь с расширенным набором прав. Используется для создания, обновления и удаления таблиц (хранилищ событий), а также получения служебной информации.
- **writer** — пользователь с правами на запись. Используется в коллекторах для записи событий в ClickHouse.
- **reader** — пользователь с правами только на чтение. Используется для выполнения поисковых запросов.

При обновлении сервера ClickHouse пользователям owner, writer и reader необходимо заново выдать требуемые права:

- Для сервера ClickHouse в **кластере Kubernetes** переназначение прав выполняется автоматически.
- Для **выделенного сервера ClickHouse** необходимо переназначить права вручную с помощью скрипта ниже. Выполните скрипт на любом экземпляре кластера ClickHouse:

```
-- Выдать необходимые права пользователю owner.
GRANT ON CLUSTER '{cluster}' ALL on default.* TO 'owner';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'owner';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO 'owner';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO 'owner';

-- Выдать необходимые права пользователю writer.
GRANT ON CLUSTER '{cluster}' SELECT ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' INSERT ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' ALTER UPDATE ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' ALTER DELETE ON default.* TO 'writer';
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'writer';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO 'writer';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO 'writer';

-- Выдать необходимые права пользователю reader.
GRANT ON CLUSTER '{cluster}' SELECT ON default.* TO 'reader';
```

```
GRANT ON CLUSTER '{cluster}' REMOTE on *.* TO 'reader';
GRANT ON CLUSTER '{cluster}' CLUSTER on *.* TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.disks TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.storage_policies TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.query_log TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.columns TO 'reader';
GRANT ON CLUSTER '{cluster}' SELECT ON system.parts_columns TO 'reader';
```

Устранение проблем при установке или обновлении

В данном разделе приведены возможные проблемы при установке, обновлении или работе системы и способы их решения.

Longhorn: ошибка конфигурации

Версия системы: 3.0 и выше. Должен быть установлен провайдер хранилища Longhorn.

Проявление: при установке или обновлении системы возникает ошибка `Unable to attach or mount volumes: unmounted volumes=[data], unattached volumes=[kube-api-access scripts data config tz]: timed out waiting for the condition`.

Причина: некорректные настройки Longhorn или на узлах кластера не запущена служба `iscsid`.

Решение:

1. Проверьте настройки Longhorn. Для этого выполните следующие действия:
   a. Осуществите проброс портов. Для этого выполните на хосте с сетевым доступом к управляющему узлу кластера следующую команду:

      ```
      kubectl -n longhorn-system port-forward service/longhorn-frontend
      <local_port>:80
      ```

      Здесь:

      - `<local_port>` — порт на хосте, по которому будет осуществляться доступ к Longhorn.

   b. В браузере откройте URL `localhost:<local_port>`, где `<local_port>` — порт с предыдущего шага. Отобразится панель управления Longhorn.

   c. Чтобы просмотреть и отредактировать конфигурацию узлов (реплик) Longhorn, перейдите в раздел **Node**.

   d. Если количество узлов Longhorn больше, чем количество рабочих узлов кластера, удалите лишние узлы. Для этого нажмите на кнопку **Remove Node** в выпадающем меню в столбце **Operation** для каждого из лишних узлов.

   e. Для каждого узла Longhorn проверьте, не превышает ли объем выделенного ему пространства размер диска узла кластера:

      i. В выпадающем меню в столбце **Operation** нажмите на кнопку **Edit node and disks**.

ii. Если в поле **Storage Reserved** выделено больше пространства, чем размер диска узла кластера, уменьшите размер пространства и нажмите на кнопку **Save**.

Рекомендуется выделять под Longhorn 30% диска узла кластера.

2. Если исправление конфигурации Longhorn не решило проблему, проверьте, запущена ли служба `iscsid` на всех узлах кластера:

   a. На каждом узле кластера выполните команду:

   ```
   systemctl status iscsid
   ```

   b. Если статус службы отличается от `active (running)`, запустите ее:

   ```
   systemctl start iscsid
   ```

Установка и удаление агентов

В этом разделе приведены инструкции по установке и удалению агента в различных операционных системах.

- Требования к установке агента
- Установка агента
- Настройка конфигурации агента
- Работа с установленными агентами
- Подключение агента к кластеру
- Подключение агента с шифрованием
  - Добавление TLS-сертификата на агенте
  - Настройка шифрования между агентом и шлюзом
- Обновление агента
- Устранение проблем с агентами
- Удаление агента

Требования к установке агента

В данном разделе представлены минимальные технические требования для установки и функционирования агента.

- Аппаратные требования
- Программные требования

Аппаратные требования

- Для установки:
  - Объем ОЗУ: 1 ГБ.
  - Свободное место на диске: 10 ГБ.
- Для работы агента:
  - CPU: не менее 2 ядер, с минимальной нагрузкой.
  - Объем ОЗУ: не менее 2 ГБ.
  - Свободное место на диске: достаточное для временного хранения данных в случае сбоя передачи в NATS.

Программные требования

Поддерживаемые ОС:

- ОС Linux:
  - Дистрибутивы семейства RedHat:
    - Rocky Linux 8, 9;
    - CentOS 7, 8, 9;
    - Red Hat Enterprise Linux 8, 9.
  - Дистрибутивы семейства Debian:
    - Ubuntu 20.04, 22.04, 24.04;
    - Debian 11, 12.
- ОС Windows:
  - Windows Server 2012 и выше;
  - Windows 10, 11.
- macOS:
  - macOS 11 и выше.

Корректная работа агентов не гарантируется на версиях macOS, вышедших менее чем три месяца спустя после даты релиза текущей версии платформы.

Установка агента

Установка агента отличается в зависимости от используемой операционной системы:

- Установка агента в ОС Linux
- Установка агента в ОС Windows
- Установка агента в macOS

Перед установкой агента ознакомьтесь с техническими требованиями.

Установка агента в ОС Linux

Чтобы установить агент в ОС Linux:

1. При необходимости измените путь установки и путь хранения логов агента в переменных окружения `EVO_AGENT_HOME` и `EVO_AGENT_LOGS`:

   ```
   export EVO_AGENT_HOME=<install_path>
   export EVO_AGENT_LOGS=<log_path>
   ```

   Здесь:

   - `<install_path>` — полный путь установки агента.
   - `<log_path>` — полный путь хранения логов.

   По умолчанию установлены следующие пути:

   - Путь установки: `/opt/rvision/agent`
   - Путь хранения логов: `/opt/rvision/agent/logs`

2. Загрузите и запустите установочный пакет. Команда запуска зависит от используемого дистрибутива Linux:

   - Для дистрибутивов семейства Debian:

     ```
     sudo dpkg -i <installer_path>
     ```

     При возникновении ошибок зависимости выполните следующую команду для их исправления:

     ```
     sudo apt-get install -f
     ```

   - Для дистрибутивов семейства RedHat:

     ```
     sudo rpm -U <installer_path>
     ```

     При возникновении ошибок зависимости выполните следующую команду для их исправления:

     ```
     sudo yum install -y
     ```

   Здесь:

- `<installer_path>` — полный путь до установщика.

Чтобы зарегистрировать установленный агент в менеджере, запустите его.

Установка агента в ОС Windows

В ОС Windows доступны следующие способы установки агента:

- Установка агента из EXE-файла
- Установка агента из MSI-пакета

Установка агента из EXE-файла

EXE-установщик агента содержит в себе MSI-пакет, а также компонент Microsoft Visual C++, необходимый для запуска агента в ОС Windows версии ниже 10. Если данный компонент уже установлен, то для установки агента рекомендуется воспользоваться MSI-пакетом.

Чтобы установить агент в ОС Windows из EXE-файла, загрузите и запустите файл установщика с помощью команды:

```
<installer_path>
```

Здесь:

- `<installer_path>` — полный путь до установщика.

При необходимости вы можете выполнить установку агента без вывода сообщений инсталлятора, добавив в команду запуска установки параметр `/q`:

```
<installer_path> /q
```

При установке агента из EXE-файла нет возможности изменить путь хранения логов. Логи будут автоматически сохраняться в папку `%TEMP%\evo-agent_<datetime>_setup.log`.

Чтобы зарегистрировать установленный агент в менеджере, запустите его.

Установка агента из MSI-пакета

Установка агента из MSI-пакета рекомендуется при наличии установленного компонента Microsoft Visual C++. Если данный компонент отсутствует, необходимо выполнить установку агента через EXE-файл.

Чтобы установить агент в ОС Windows из MSI-пакета:

1. При необходимости измените путь установки агента в переменной окружения `RVISION_AGENT_INSTALL_PATH`.

По умолчанию агент устанавливается в папку `C:\Program Files (x86)\RVision\Agent`.

2. Загрузите и запустите установочный пакет с помощью команды:

```
msiexec /i <installer_path> /L*V '<log_path>'
```

Здесь:

- `<installer_path>` — полный путь до установщика.
- `<log_path>` — полный путь хранения логов.

Чтобы зарегистрировать установленный агент в менеджере, запустите его.

Для корректной установки агента требуется запускать установочный пакет с правами администратора. Если появляется сообщение об отсутствии DLL-файлов, скачайте их из актуальной библиотеки Microsoft Visual C++ и установите их вручную. Агент применит установленные DLL-файлы для продолжения установки.

Установка агента в macOS

Чтобы установить агент в macOS, загрузите и запустите установочный пакет с помощью команды:

```
sudo installer -pkg <installer_path> -target /
```

Здесь:

- `<installer_path>` — полный путь до установщика.

Пути установки и хранения логов хранятся в переменных `EVO_AGENT_HOME` и `EVO_AGENT_LOGS` соответственно.

По умолчанию установлены следующие пути:

- Путь установки: `/Library/R-Vision/agent`
- Путь хранения логов: `/Library/R-Vision/agent/logs`

Чтобы зарегистрировать установленный агент в менеджере, запустите его.

Настройка конфигурации агента

Конфигурация агента настраивается в файле rpoint.conf.

Файл конфигурации rpoint.conf создается автоматически при установке агента и хранится в подкаталоге `etc` каталога установки.

Пути хранения файла конфигурации по умолчанию:

- Для ОС Linux: `/opt/rvision/agent/etc/`
- Для ОС Windows: `C:\Program Files (x86)\RVision\Agent\etc\`
- Для macOS: `/Library/R-Vision/agent/etc/`

Структура файла конфигурации

Файл конфигурации rpoint.conf содержит следующие секции настроек:

- `log` — настройки логирования.
- `nats` — настройки подключения к локальному NATS-серверу.
- `nats_server` — настройки подключения к внешнему NATS-серверу.
- `internal_audit` — настройки аудита.
- `inventory` — настройки сбора инвентарной информации.

▼ *Пример содержимого файла конфигурации rpoint.conf*

```
{
  "log": {
    "file_path": "logs/rpoint.log",
    "level": "info",
    "rotation_size": 16384,
    "target_file_name": "rpoint_%N.log"
  },
  "nats": {
    "local": {
      "connection": {
        "url": "nats://127.0.0.1:4111"
      }
    }
  },
  "nats_server": {
    "http_port": 8222,
    "port": 4111,
    "jetstream": {
      "max_memory_store": 1073741824,
      "max_file_store": 10737418240
    },
    "leafnodes": {
        "remotes": [
            {
              "url": "nats://worker01.rpoint.local:32000"
            }
        ]
    }
  },
  "internal_audit": {
    "monitoring_interval": 60,
    "local_queue_path": "queue",
```

```
    "send_timeout": 10000
  },
  "inventory": {
    "monitoring_interval": 86400
  }
}
```

Секция `log`

В данной секции задаются настройки логирования.

Параметр	Описание	Пример
`file_path`	Путь к файлу логирования на устройстве.	`"file_path": "logs/rpoint.log",`
`level`	Уровень логирования. Доступные значения: • `trace`; • `debug`; • `info`; • `warning`; • `error`; • `fatal`.	`"level": "info",`
`rotation_size`	Размер файла, по достижении которого выполняется ротация логов. Задается, если для ротации указан критерий по размеру файла.	`"rotation_size": 16384,`
`time_rotation`	Время ротации логов. Задается, если для ротации указан критерий по времени.	`"time_rotation": { "hours": 24, "minutes": 60, "seconds": 60 },`
`target_file_name`	Имя файла для ротации логов.	`"target_file_name": "rpoint_%N.log"`

Секция `nats`

В данной секции задаются настройки подключения к локальному NATS-серверу.

Параметр	Описание	Пример

`local`	Данные для подключения к локальному NATS-серверу:
	• `url` — URL-адрес локального NATS-сервера в формате `nats://<addr>:<port>`, где:
	◦ `<addr>` — адрес подключения к серверу.
	◦ `<port>` — порт подключения к серверу.
		`"local": {`
		`  "connection": {`
		`  "url": "nats://127.0.0.1:4111"`
		`  }`
		`}`

Секция `nats_server`

В данной секции задаются настройки подключения к внешнему NATS-серверу.

Параметр	Описание	Пример
`http_port`	Порт для мониторинга подключения к внешнему NATS-серверу.	`"http_port": 8222,`
`port`	Порт локального NATS-сервера.	`"port": 4111,`

`jetstream`	Параметры брокера сообщений NATS JetStream:
	• `max_memory_store` — максимальный объем занимаемой памяти в байтах.
	• `max_file_store` — максимальный объем файла на дисковом пространстве в байтах.
		`"jetstream": {`
		`  "max_memory_store": 1073741824,`
		`  "max_file_store": 10737418240`
		`},`

Параметр	Описание	Пример
`leafnodes`	Настройки шлюзов для подключения. Для каждого шлюза без TLS-шифрования в поле `url` указывается URL-адрес в формате `nats://<addr>:<port>`. Для каждого шлюза с TLS-шифрованием адрес указывается в формате `tls://<addr>:<port>`. Здесь: • `<addr>` — адрес подключения к шлюзу. • `<port>` — порт подключения к шлюзу.	`"leafnodes": { "remotes": [ { "url": "nats://worker01.rpoint.local:32000" } ] }`

Секция `internal_audit`

В данной секции задаются настройки аудита.

Параметр	Описание	Пример
`monitoring_interval`	Интервал выполнения задачи аудита в секундах.	`"monitoring_interval": 60,`
`local_queue_path`	Путь к файлам хранения локальной очереди.	`"local_queue_path": "queue",`
`send_timeout`	Таймаут на выполнение запросов аудита в секундах.	`"send_timeout": 10000`

Секция `inventory`

В данной секции задаются настройки сбора инвентарной информации об агенте.

Параметр	Описание	Пример
`monitoring_interval`	Интервал сбора инвентарной информации в секундах.	`"monitoring_interval": 86400`

Работа с установленными агентами

Над установленным агентом доступны следующие операции:

- Запуск агента

- Перезапуск агента
- Остановка агента
- Запрос статуса агента

Запуск агента

Для запуска агента необходимо обладать правами администратора или root-пользователя.

Способ запуска агента зависит от используемой ОС:

- В ОС Linux:

```
sudo ${EVO_AGENT_HOME}/bin/rpoint-launcher start
```

- В ОС Windows:
  - Первый способ запуска агента:

```
Start-Service -Name RvisionAgent
```

  - Второй способ запуска агента:

```
& '<install_path>\bin\rpoint-launcher.exe' start
```

    Здесь:

    - `<install_path>` — полный путь установки агента.

Если перед установкой агента была задана переменная `RVISION_AGENT_INSTALL_PATH`, то вместо указания полного пути установки можно обратиться к этой переменной:

```
& '${RVISION_AGENT_INSTALL_PATH}\bin\rpoint-launcher.exe' start
```

- В macOS:

```
sudo ${EVO_AGENT_HOME}/bin/rpoint-launcher start
```

При запуске агента осуществляется подключение к внешнему NATS-серверу, указанному в конфигурационном файле rpoint.conf, а также выполняется сбор его инвентарной информации.

При необходимости вы можете указать адрес подключения к внешнему NATS-серверу явно через параметр `--gateway-url`. Например:

- В ОС Linux:

```
sudo ${EVO_AGENT_HOME}/bin/rpoint-launcher start --gateway
-url=nats://<addr>:<port>
```

- В ОС Windows:

```
& '<install_path>\bin\rpoint-launcher.exe' start --gateway
-url=nats://<addr>:<port>
```

- В macOS:

```
sudo ${EVO_AGENT_HOME}/bin/rpoint-launcher start --gateway
-url=nats://<addr>:<port>
```

Здесь:

- `<addr>` — адрес подключения к шлюзу.
- `<port>` — порт подключения к шлюзу.

Перезапуск агента

Способ перезапуска агента зависит от используемой ОС:

- В ОС Linux:

```
${EVO_AGENT_HOME}/bin/rpoint-launcher restart
```

- В ОС Windows:
  - Первый способ перезапуска агента:

```
Restart-Service -Name RvisionAgent
```

  - Второй способ перезапуска агента:

```
& '<install_path>\bin\rpoint-launcher.exe' restart
```

Здесь:

- `<install_path>` — полный путь установки агента.

Если перед установкой агента была задана переменная `RVISION_AGENT_INSTALL_PATH`, то вместо указания полного пути установки

можно обратиться к этой переменной:

```
& '${RVISION_AGENT_INSTALL_PATH}\bin\rpoint-launcher.exe'
restart
```

- В macOS:

```
${EVO_AGENT_HOME}/bin/rpoint-launcher restart
```

При перезапуске агента выполняется сбор его инвентарной информации.

Остановка агента

Способ остановки агента зависит от используемой ОС:

- В ОС Linux:

```
${EVO_AGENT_HOME}/bin/rpoint-launcher stop
```

- В ОС Windows:
  - Первый способ остановки агента:

```
Stop-Service -Name RvisionAgent
```

  - Второй способ остановки агента:

```
& '<install_path>\bin\rpoint-launcher.exe' stop
```

Здесь:

- `<install_path>` — полный путь установки агента.

Если перед установкой агента была задана переменная `RVISION_AGENT_INSTALL_PATH`, то вместо указания полного пути установки можно обратиться к этой переменной:

```
& '${RVISION_AGENT_INSTALL_PATH}\bin\rpoint-launcher.exe' stop
```

- В macOS:

```
${EVO_AGENT_HOME}/bin/rpoint-launcher stop
```

Остановка процессов агента

Помимо полной остановки агента, можно также останавливать его конкретные процессы. Способ остановки процессов зависит от используемой ОС:

- В ОС Linux и macOS:

```
sudo pkill -9 <process_name>
```

- В ОС Windows:

```
Stop-Process -Name "<process_name>" -Force -ErrorAction SilentlyContinue
```

Здесь:

- `<process_name>` — имя процесса, который необходимо остановить.

- Если попытаться остановить любой процесс, кроме rpoint-launcher, то этот процесс будет автоматически перезапущен.
- Если попытаться остановить процесс rpoint-launcher, то дальнейшее поведение процессов будет зависеть от сервиса WatchDog:
  - Если сервис WatchDog активен, то процесс rpoint-launcher будет автоматически перезапущен.
  - Если сервис WatchDog неактивен, то процесс rpoint-laucnher не будет перезапущен. В этом случае остальные процессы агента могут быть также успешно остановлены.

Запрос статуса агента

Способ получения информации о текущем статусе агента зависит от используемой ОС:

- В ОС Linux:

```
${EVO_AGENT_HOME}/bin/rpoint-launcher status
```

- В ОС Windows:
  - Первый способ запроса статуса агента:

```
Get-Service -Name RvisionAgent
```

  - Второй способ запроса статуса агента:

```
& '<install_path>\bin\rpoint-launcher.exe' status
```

Здесь:

- `<install_path>` — полный путь установки агента.

Если перед установкой агента была задана переменная `RVISION_AGENT_INSTALL_PATH`, то вместо указания полного пути установки можно обратиться к этой переменной:

```
& '${RVISION_AGENT_INSTALL_PATH}\bin\rpoint-launcher.exe' status
```

- В macOS:

```
${EVO_AGENT_HOME}/bin/rpoint-launcher status
```

В результате выполнения команды будет выведена информация о текущем статусе агента.

Пример 6. Пример информации о статусе агента

```
[{"name":"nats-server","status":"running"},{"name":"configuration-
server","status":"running"},{"name":"vm","reason":"Disabled by config","status":"not
running"}]
```

Если в ОС Windows статус запрашивается с помощью команды `Get-Service`, то результат вывода информации отличается.

Пример 7. Пример информации о статусе агента для команды `Get-Service`

```
Status   Name               DisplayName
-------  --------           -----------
Running  RvisionAgent       EVO Agent Service
```

После установки и запуска агента модули `nats-server` и `configuration-server` должны иметь статус `running`, например:

```
[{"name":"nats-server","status":"running"},{"name":"configuration-
server","status":"not running"},...]
```

Если статус хотя бы одного из модулей отличается от `running`, необходимо переустановить агент.

Подключение агента к кластеру

Чтобы подключить агент к основному кластеру или сателлиту, выполните следующие действия:

1. Создайте и настройте шлюз в кластере.

2. Задайте настройки подключения агента к шлюзу.

В данной инструкции рассматривается подключение агента без шифрования соединения. Если необходимо настроить агент с шифрованием, обратитесь к разделу Настройка шифрования между агентом и шлюзом.

Создание и настройка шлюза

Чтобы настроить шлюз в кластере:

1. Создайте шлюз. При создании укажите следующие параметры:
   - IP- или FQDN-адрес для подключения к шлюзу.

Адреса, используемые для подключения к шлюзу, зависят от типа кластера, в котором создается шлюз.

- Если шлюз создается в пространстве центрального управляющего кластера, используйте IP-адреса любого узла центрального кластера.
- Если шлюз создается в пространстве кластера, являющегося сателлитом, используйте IP-адреса любого узла сателлита.

FQDN-адрес можно использовать, только если в инфраструктуре существует DNS-сервер и администратор настроил прямую зону (forward zone). Проверить, настроена ли такая зона, можно с помощью утилиты nslookup:

```
nslookup <domain_name>
```

Здесь:

- `<domain_name>` — доменное имя.

Обратные DNS-запросы для определения FQDN-адресов по IP-адресам возможны, если на DNS-сервере настроена обратная зона (reverse zone). В ней хранятся записи, сопоставляющие IP-адреса с доменными именами. Проверить, настроена ли такая зона, можно также с помощью утилиты nslookup:

```
nslookup <IP_address>
```

Здесь:

- `<IP_address>` — IP-адрес узла.

Если обратная зона не настроена, свяжитесь с сетевым администратором.

Если ваш компьютер имеет доступ к кластеру с установленным файлом конфигурации kubeconfig, вы можете получить сведения обо всех IP-адресах узлов кластера. Для сбора IP-адресов всех узлов в кластере и их вывода в

виде строки используйте команду:

```
kubectl get node -o json
--output='jsonpath={.items[*].status.addresses[0].address}'
```

- Порт для подключения к шлюзу. Допустимые значения: 30 000—32 767.

Порт должен быть уникальным в рамках инсталляции системы.

- Максимальный размер буфера в МБ для хранения событий ИБ.

Эффективный размер буфера будет равен 80% от указанного значения.

2. Запустите шлюз. Шлюз станет доступен для подключения к нему агентов.

Подключение агента к шлюзу

Чтобы подключить агент к шлюзу:

1. Установите агент.
2. Настройте сетевой доступ к шлюзу по необходимому порту. Для этого задайте настройки подключения к шлюзу в параметре `leafnodes` в секции `nats_server` файла конфигурации rpoint.conf.

▼ *Особенности задания параметра* `leafnodes`

Для каждого шлюза без TLS-шифрования в поле `url` указывается URL-адрес в формате `nats://<addr>:<port>`.

Для каждого шлюза с TLS-шифрованием адрес указывается в формате `tls://<addr>:<port>`.

Здесь:

- `<addr>` — адрес подключения к шлюзу.
- `<port>` — порт подключения к шлюзу.

```
"leafnodes": {
  "remotes": [
    {
      "url": "nats://worker01.rpoint.local:32000"
    }
  ]
}
```

3. Запустите агент.

При необходимости вы можете пропустить шаг 2 и запустить агент сразу с необходимыми настройками подключения к шлюзу. Для этого к команде

запуска необходимо добавить параметр `--gateway-url`. Например:

- В ОС Linux:

```
sudo ${EVO_AGENT_HOME}/bin/rpoint-launcher start --gateway
-url=nats://<addr>:<port>
```

- В ОС Windows:

```
& '<install_path>\bin\rpoint-launcher.exe' start --gateway
-url=nats://<addr>:<port>
```

- В macOS:

```
sudo ${EVO_AGENT_HOME}/bin/rpoint-launcher start --gateway
-url=nats://<addr>:<port>
```

Здесь:

- `<addr>` — адрес подключения к шлюзу.
- `<port>` — порт подключения к шлюзу.

Подключение агента с шифрованием

Шифрование соединения обеспечивает безопасную передачу данных между агентами и шлюзами. Шифрование осуществляется через протокол TLS (Transport Layer Security).

Настройка подключения агента к кластеру с шифрованием данных состоит из следующих этапов:

1. Добавление TLS-сертификата на агенте
2. Настройка шифрования между агентом и шлюзом

Добавление TLS-сертификата на агенте

Для шифрования соединения между агентами и шлюзами необходимо использовать TLS-сертификат.

TLS-сертификат — это цифровой документ, позволяющий использовать протокол безопасного соединения TLS (Transport Layer Security). Наличие TLS-сертификата предоставляет следующие гарантии:

- Ресурс, на котором расположен агент, является подлинным и не принадлежит злоумышленникам.
- Передаваемая информация защищена от злоумышленников благодаря установке зашифрованного соединения между агентом и шлюзом.

Чтобы добавить TLS-сертификат на агент:

1. Сгенерируйте TLS-сертификат.
2. Установите корневой сертификат на компьютере агента.

Добавление сертификатов выполняется после установки агента.

Генерация TLS-сертификата

В представленной инструкции в качестве примера используются файлы со следующими названиями:

- **ca.key** — файл приватного ключа корневого сертификата;
- **ca.crt** — файл корневого сертификата;
- **server.ext** — файл конфигурации TLS-сертификата;
- **server.key** — файл приватного ключа TLS-сертификата;
- **server.csr** — файл запроса на выпуск TLS-сертификата;
- **server.crt** — файл TLS-сертификата.

Чтобы сгенерировать TLS-сертификат, выполните на компьютере с установленным агентом следующие действия:

1. Создайте файл **server.ext** со следующей конфигурацией TLS-сертификата:

```
authorityKeyIdentifier=keyid,issuer

basicConstraints=CA:FALSE

keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment

subjectAltName = @alt_names

[alt_names]
DNS.1 = <domain_name_1>
...
DNS.N = <domain_name_N>
```

Здесь:

- <domain_name_1>, …, <domain_name_N> — доменные имена шлюзов, с которыми необходимо настроить безопасное соединение.

▼ *Пример конфигурации TLS-сертификата*

```
authorityKeyIdentifier=keyid,issuer

basicConstraints=CA:FALSE

keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
```

```
subjectAltName = @alt_names

[alt_names]
DNS.1 = worker01.rpoint.local
DNS.2 = worker02.rpoint.local
DNS.3 = worker03.rpoint.local
```

2. Создайте приватный ключ **ca.key** для корневого сертификата:

```
openssl genrsa -out ca.key 4096
```

3. Создайте самоподписанный корневой сертификат **ca.crt**:

```
openssl req -x509 -new -nodes -key ca.key -sha256 -days 3650 -out ca.crt
```

4. Создайте приватный ключ **server.key** для TLS-сертификата:

```
openssl genrsa -out server.key 2048
```

5. Создайте запрос на выпуск сертификата **server.csr**:

```
openssl req -new -key server.key -out server.csr
```

6. Подпишите запрос **server.csr** с помощью корневого сертификата **ca.crt**:

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out
server.crt -days 825 -sha256 -extfile server.ext
```

   В результате будет сгенерирован файл TLS-сертификата **server.crt**.

Установка корневого сертификата на компьютере с агентом

В представленной инструкции в качестве примера используется файл корневого сертификата **ca.crt**.

Для TLS-шифрования соединения между агентом и шлюзом необходимо установить корневой сертификат, созданный на этапе Генерация TLS-сертификата, на компьютере агента.

Процесс установки корневого сертификата отличается в зависимости от используемой ОС:

- Установка корневого сертификата в ОС Linux
- Установка корневого сертификата в ОС Windows
- Установка корневого сертификата в macOS

Установка корневого сертификата в ОС Linux

В инструкции для установки сертификата в ОС Linux используется пакет ca-certificates:

- Пакет для Debian-подобных дистрибутивов
- Пакет для RedHat-подобных дистрибутивов

Чтобы установить корневой сертификат в ОС Linux:

- Для Debian-подобных дистрибутивов:
  1. Скопируйте файл корневого сертификата в каталог `/usr/local/share/ca-certificates/`:

     ```
     cp <path_to_cert> /usr/local/share/ca-certificates/
     ```

     Здесь:

     - `<path_to_cert>` — полный путь до файла корневого сертификата **ca.crt**.
  2. Выполните команду:

     ```
     update-ca-certificates
     ```

     Корневой сертификат будет добавлен в список доверенных.
- Для RedHat-подобных дистрибутивов:
  1. Скопируйте файл корневого сертификата в каталог `/etc/pki/ca-trust/source/anchors/`:

     ```
     cp <path_to_cert> /etc/pki/ca-trust/source/anchors/
     ```

     Здесь:

     - `<path_to_cert>` — полный путь до файла корневого сертификата **ca.crt**.
  2. Выполните команду:

     ```
     update-ca-trust
     ```

     Корневой сертификат будет добавлен в список доверенных.

Установка корневого сертификата в ОС Windows

Чтобы установить корневой сертификат в ОС Windows:

1. Откройте диспетчер сертификатов. Для этого выполните следующие действия:
   a. Нажмите сочетание клавиш WIN+R. Откроется окно **Выполнить**.

b. Введите в окне `certlm.msc` и нажмите **ОК**.

2. Перейдите в раздел **Доверенные корневые центры сертификации** → **Сертификаты** с помощью дерева разделов в диспетчере сертификатов.
3. Выполните команду **Все задачи** → **Импорт** в контекстном меню раздела **Доверенные корневые центры сертификации** → **Сертификаты** в дереве разделов. Откроется мастер импорта сертификатов.
4. Нажмите **Далее** на стартовой странице мастера.
5. Выберите файл корневого сертификата **ca.crt** на странице выбора сертификата и нажмите **Далее**.
6. Удостоверьтесь, что на странице выбора хранилища выбран вариант **Поместить все сертификаты в следующее хранилище** и выбрано хранилище **Доверенные корневые центры сертификации**. Нажмите **Далее**.
7. Проверьте корректность указанных параметров на странице завершения импорта и нажмите **Готово**. Корневой сертификат будет добавлен в список доверенных.

Вы также можете установить корневой сертификат следующим способом:

1. Дважды нажмите на файл сертификата **ca.crt**. Откроется окно свойств сертификата.
2. На вкладке **Общее** нажмите на кнопку **Установить сертификат**. Откроется мастер импорта сертификатов.
3. На стартовой странице мастера выберите расположение хранилища **Локальный компьютер** и нажмите **Далее**.
4. На странице выбора хранилища выполните следующие действия:
   a. Выберите вариант **Поместить все сертификаты в следующее хранилище**.
   b. Выберите хранилище **Доверенные корневые центры сертификации**.
   c. Нажмите **Далее**.
5. Проверьте корректность указанных параметров на странице завершения импорта и нажмите **Готово**. Корневой сертификат будет добавлен в список доверенных.

Установка корневого сертификата в macOS

Чтобы установить корневой сертификат в macOS:

1. Откройте приложение **Связка ключей**.
2. Откройте приложение **Finder** и найдите в нем файл корневого сертификата **ca.crt**. Перетащите найденный файл в окно приложения **Связка ключей**.
3. Перейдите на вкладку **Сертификаты** в приложении **Связка ключей**.
4. Дважды нажмите на добавленный корневой сертификат. Откроется окно сведений о сертификате.
5. Раскройте группу параметров **Доверие** и для каждого параметра выберите вариант **Всегда доверять**. Корневой сертификат будет добавлен в список доверенных.

Вы также можете установить корневой сертификат в macOS с помощью команды:

```
sudo security add-trusted-cert -d -r trustRoot -k
/Library/Keychains/System.keychain <path_to_cert>
```

Здесь:

- `<path_to_cert>` — полный путь до файла корневого сертификата **ca.crt**.

Настройка шифрования между агентом и шлюзом

Чтобы настроить шифрование соединения между агентом и шлюзом:

1. Создайте секрет.
2. Создайте и настройте шлюз.
3. Настройте шифрование на агенте.

Создание секрета

Чтобы создать секрет с подготовленным TLS-сертификатом:

1. Перейдите в раздел **Ресурсы** → **Секреты**. Система отобразит сведения об имеющихся секретах.
2. Нажмите на кнопку **Создать** (＋). Отобразится окно создания секрета.
3. Введите название секрета.
4. При необходимости введите описание секрета.
5. Выберите из выпадающего списка тип секрета **Контейнер PKCS**.
6. Задайте настройки выбранного типа секрета:
   a. Загрузите файл сертификата и файл приватного ключа, подготовленные на этапе Добавление TLS-сертификата на агенте.
   b. Оставьте поле **Пароль для приватного ключа** пустым.

   Система не поддерживает использование приватных ключей с паролем для TLS-шифрования соединения.
7. Нажмите на кнопку **Создать**. Система создаст секрет и отобразит соответствующее уведомление. Новый секрет появится в списке раздела **Ресурсы** → **Секреты**.

Создание и настройка шлюза

Чтобы настроить шифрование соединения со шлюзом:

1. Создайте шлюз. При создании задайте следующие параметры:
   a. Укажите IP- или FQDN-адрес для подключения к шлюзу.

   Адрес шлюза с шифрованием должен совпадать с адресом, указанном в

файле TLS-сертификата.

b. Укажите порт для подключения к шлюзу. Допустимые значения: 30 000—32 767.

Порт должен быть уникальным в рамках инсталляции системы.

c. Задайте максимальный размер буфера в МБ для хранения событий ИБ.

Эффективный размер буфера будет равен 80% от указанного значения.

d. Переведите переключатель **Включить TLS** в активное положение и выберите секрет, созданный на шаге Создание секрета.

2. Запустите шлюз. Шлюз станет доступен для подключения к нему агентов.

Настройка шифрования на агенте

Чтобы подключить агент к шлюзу с шифрованием соединения:

1. Установите агент.
2. Настройте сетевой доступ к шлюзу по необходимому порту. Для этого задайте настройки подключения к шлюзу в параметре `leafnodes` в секции `nats_server` файла конфигурации rpoint.conf.

▼ *Особенности задания параметра* `leafnodes` *с шифрованием*

Для каждого шлюза без TLS-шифрования в поле `url` указывается URL-адрес в формате `nats://<addr>:<port>`.

Для каждого шлюза с TLS-шифрованием адрес указывается в формате `tls://<addr>:<port>`.

Здесь:

- `<addr>` — адрес подключения к шлюзу.
- `<port>` — порт подключения к шлюзу.

```
"leafnodes": {
  "remotes": [
    {
      "url": "tls://worker01.rpoint.local:32000"
    }
  ]
}
```

3. Запустите агент.

Обновление агента

Обновление агента отличается в зависимости от используемой операционной системы:

- Обновление агента в ОС Linux
- Обновление агента в ОС Windows
- Обновление агента в macOS

Если при обновлении агента произошел сбой, выполните следующие действия:

1. Удалите агент.
2. Установите агент заново.

Обновление агента в ОС Linux

Чтобы обновить агент в ОС Linux, загрузите и запустите установочный пакет. Команда запуска зависит от используемого дистрибутива Linux:

- Для дистрибутивов семейства Debian:

  ```
  sudo dpkg -i <installer_path>
  ```

  При возникновении ошибок зависимости выполните следующую команду для их исправления:

  ```
  sudo apt-get install -f
  ```

- Для дистрибутивов семейства RedHat:

  ```
  sudo rpm -U <installer_path>
  ```

  При возникновении ошибок зависимости выполните следующую команду для их исправления:

  ```
  sudo yum install -y
  ```

  Здесь:

  - `<installer_path>` — полный путь до установщика.

Обновление агента в ОС Windows

Чтобы обновить агент в ОС Windows, загрузите и запустите установочный пакет с помощью команды:

```
msiexec /i <installer_path> /L*V '<log_path>'
```

Здесь:

- `<installer_path>` — полный путь до установщика.

- `<log_path>` — полный путь хранения логов.

Обновление агента в macOS

Чтобы обновить агент в macOS, загрузите и запустите установочный пакет с помощью команды:

```
sudo installer -pkg <installer_path> -target /
```

Здесь:

- `<installer_path>` — полный путь до установщика.

Устранение проблем с агентами

В данном разделе приведены возможные проблемы при установке или обновлении агента и способы их решения.

Некорректные статусы модулей `nats-server` и `configuration-server`

Проблема: после установки и запуска агента статусы модулей `nats-server` и `configuration-server` отличаются от `running`, например:

```
[{"name":"nats-server","status":"running"},{"name":"configuration-
server","reason":"Down/Starting..","status":"not running"},...]
```

Решение: переустановите агент. Для этого:

1. Удалите агент.
2. Перезапустите машину.
3. Заново установите и запустите агент.

Установленный агент не отображается в менеджере агентов

Проблема: после установки и запуска агент не отображается в менеджере агентов.

Решение:

1. Если система отображает ошибку вида `Error trying to connect as leafnode to remote server`:
   a. Проверьте подключение компьютера к сети. Если подключение отсутствует, восстановите его.
   b. Проверьте доступность шлюза, указанного в конфигурационном файле rpoint.conf. Если указаны некорректные настройки подключения (например, заданы закрытые порты), исправьте их.
2. Откройте файл `logs/nats.log` и найдите последнее сообщение.
   - Если сообщение содержит запись вида `10.99.66.43:32422 - lid:29 - JetStream using domains: local "agent-19395b65-6bdd-5a00-9970-47668a6ce92c"`, подключение к NATS прошло успешно. Вероятнее всего, проблема в конфигурации пространства основного кластера или

сателлита.

Для решения проблемы удостоверьтесь в корректности заданных настроек подключения к пространству:

- Если настройки некорректны, исправьте их.
- Если настройки корректны, но ошибка сохраняется, обратитесь в службу поддержки по адресу support@rvision.ru.

◦ Если сообщение содержит запись вида `Error trying to connect as leafnode to remote server "ep-worker03.rpoint.test.dev.rvision.local:32422" (attempt 1): dial tcp 10.99.66.43:32422: connect: connection refused`, указанный в сообщении порт недоступен на оборудовании.

Для решения проблемы обратитесь к системному администратору с запросом на открытие порта.

3. Если на предыдущих шагах проблемы не были выявлены и подключение агента прошло успешно, возможно, IP-адреса, к которым должны подключаться агенты, не указаны в файле `etc/nats-leafnode.conf`.

Для решения проблемы:

a. Откройте файл конфигурации агента rpoint.conf.

b. Проверьте, правильно ли указаны в файле IP-адреса, к которым должны подключаться агенты.

Если IP-адреса указаны неправильно, укажите верные IP-адреса в файле rpoint.conf.

Если IP-адреса указаны правильно, но данные тенантов и шлюзов не отображаются в файле nats-leafnode.conf, проверьте корректность этих данных в менеджере агентов:

- Если данные тенантов и шлюзов некорректны, исправьте их.
- Если данные корректны, ошибка, вероятнее всего, связана с самим агентом. Обратитесь в службу поддержки по адресу support@rvision.ru.

Сбой при обновлении агента

Проблема: во время обновления агента произошел сбой.

Решение:

1. Удалите агент.
2. Установите агент заново.

Удаление агента

Удаление установленного агента отличается в зависимости от используемой операционной системы:

- Удаление агента в ОС Linux

- Удаление агента в ОС Windows
- Удаление агента в macOS

Удаление агента в ОС Linux

Команда удаления агента в ОС Linux зависит от используемого дистрибутива:

- Для дистрибутивов семейства Debian:

```
sudo dpkg -r evo-agent
```

- Для дистрибутивов семейства RedHat:

```
sudo rpm -e evo-agent
```

Удаление агента в ОС Windows

Удалить агент в ОС Windows можно следующими способами:

- Удаление агента через окно параметров:
  1. Откройте меню **Пуск**.
  2. Откройте окно **Параметры**.
  3. Перейдите в раздел **Приложения** → **Приложения и возможности**. Отобразится список приложений в системе.
  4. Найдите в списке приложений evo-agent и нажмите на него. Отобразится панель действий над приложением.
  5. Нажмите на кнопку **Удалить** и подтвердите удаление.
- Удаление агента через PowerShell:
  1. Запустите PowerShell от имени администратора.
  2. Выполните команду:

```
Uninstall-Package Evo-Agent
```

Удаление агента в macOS

Чтобы удалить агент в macOS:

1. Остановите процессы агента:

```
sudo bin/rpoint-launcher stop
```

Для проверки остановки процессов используйте команду:

```
pgrep -x 'rpoint-analyzer' 'nats' 'rpoint-launcher' 'rpoint-
configuration-server' || echo 'stopped'
```

Если процессы остановлены успешно, команда выведет `stopped`, в противном случае — отобразит сведения о неостановленных процессах.

2. Удалите приложение агента:

```
sudo rm -rf /Applications/evo-agent.app
```

Для проверки удаления приложения используйте команду:

```
[ ! -d /Applications/evo-agent.app ] && echo "deleted" || echo
"exists"
```

Если приложение удалено успешно, команда выведет `deleted`, в противном случае — `exists`.

3. Если агент был установлен через pkgutil, удалите пакет агента:

```
sudo pkgutil --forget $(pkgutil --pkgs | grep "evo-agent")
```

Для проверки удаления пакета используйте команду:

```
pkgutil --pkgs
```

Если пакет удален успешно, в выводе команды будет отсутствовать `evo-agent`.

4. Удалите папку агента:

```
sudo rm -rf /Library/R-Vision/agent
```

Для проверки удаления папки используйте команду:

```
[ ! -d /Library/R-Vision/agent ] && echo "deleted" || echo "exists"
```

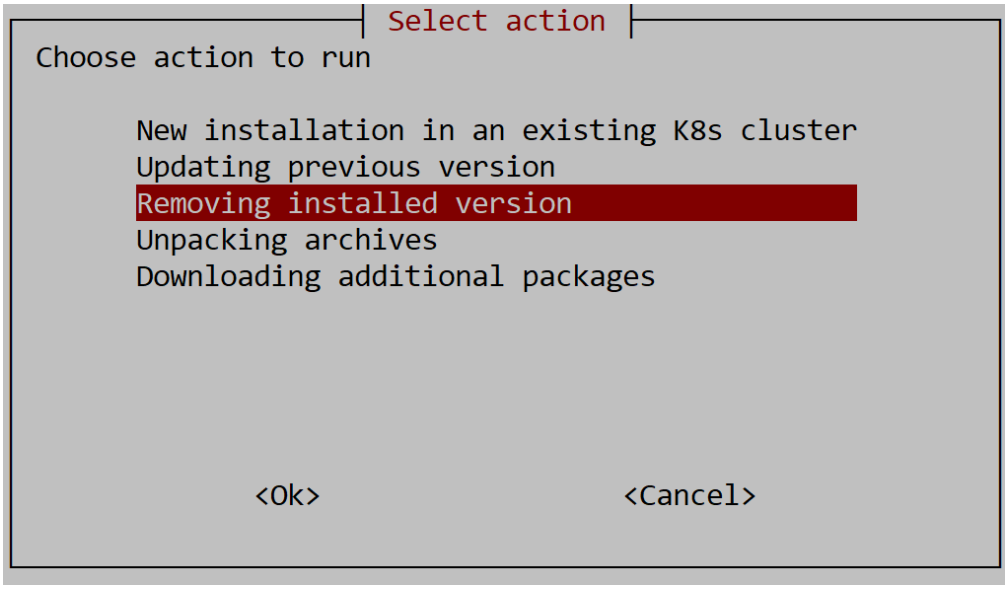
Если папка удалена успешно, команда выведет `deleted`, в противном случае — `exists`.

Удаление системы

В данном разделе описаны действия по удалению системы или сателлита.

Чтобы удалить систему:

1. Запустите установщик и выберите режим установки **Removing installed version**.



2. Выберите пространство имен (namespace), в котором установлена система.
3. Если требуется удалить данные и конфигурации Kubernetes, выберите опцию **Remove configurations and data**. Будут удалены следующие данные:
   - конфигурации дискового пространства (Persistent Volume Claims);
   - карты конфигурации Kubernetes (config maps);
   - секреты.
4. Для подтверждения удаления системы введите `yes`.

Будут удалены все развернутые helm-релизы из целевого пространства имен.

Администрирование системы

В данном разделе представлена информация о настройке и администрировании системы.

Содержание раздела:

- Добавление SSL-сертификата
  - Добавление самоподписанного сертификата в список доверенных
- Работа с кластером
  - Подключение к кластеру
  - Мониторинг кластера
    - Состояние кластера
    - Создание пользовательских дашбордов в Grafana
    - Анализ работоспособности сервисов кластера
    - Анализ работоспособности контрольной плоскости кластера
    - Анализ состояния сателлита
    - Просмотр журнала событий кластера
    - Просмотр состояния подов кластера
    - Список подов сервисов
    - Настройка оповещений
      - Настройка оповещений в Telegram
      - Настройка оповещений по электронной почте
    - Работа с метриками Prometheus
  - Управление кластером
    - Управление узлами кластера
    - Работа с подом collector-manager
    - Увеличение размера тома для подов
    - Резервное копирование и восстановление PostgreSQL
    - Резервное копирование хранилища etcd
    - Получение идентификатора сервиса в NATS Service Discovery
    - Работа с объектным хранилищем NATS Object Store
  - Работа с паролями в кластере
  - Изменение доменного имени установленного экземпляра системы
- Настройка системы
  - Настройка интеграции с продуктами R-Vision
  - Настройка формата даты
  - Изменение пула адресов в MetalLB

    - Настройка времени хранения запросов
- Управление лицензией
- Управление пространствами
    - Работа с пространствами
    - Работа с сервисами
- Управление доступом
    - Тенанты
    - Пользователи
    - Группы пользователей
    - Роли
        - Стандартные роли
    - Политики входа и пароля
        - Настройка времени жизни для токенов доступа
    - LDAP-соединения
        - Основные пользовательские атрибуты Active Directory
    - Политики управления доступом к событиям
- Настройка агентов
    - Настройки менеджера агентов
    - Сбор инвентарной информации
    - Сбор событий на агентах
- Аудит
    - Аудит источников событий
        - Политики аудита источников событий
        - Настройка конвейера для аудита коллектора
    - Аудит системных событий
        - Модель события аудита
        - Отслеживание действий суперадминистраторов
        - Отслеживание действий над объектами
- Мониторинг RQL-запросов и управление ресурсами
- Устранение проблем при эксплуатации системы

Добавление SSL-сертификата

Для работы системы по протоколу безопасного соединения HTTPS необходимо использовать SSL-сертификат.

SSL-сертификат — это цифровой документ, позволяющий использовать протокол безопасной связи

SSL (Secure Sockets Layer). Применение SSL-сертификата предоставляет следующие гарантии для пользователей:

- Ресурс, на котором расположена система, является подлинным и не принадлежит злоумышленникам.
- Передаваемая информация защищена от злоумышленников благодаря установке зашифрованного соединения между ресурсом и браузером пользователя.

Для работы с системой вы можете добавить доверенный или самоподписанный сертификат.

Добавление сертификатов выполняется после установки R-Vision.

Добавление доверенного SSL-сертификата

Доверенный SSL-сертификат — это сертификат, заверенный подписью удостоверяющего центра (УЦ). Такой сертификат показывает, что ресурс был проверен УЦ на предмет подлинности и безопасности работы.

В представленной инструкции в качестве примера используются файлы со следующими названиями:

- **ssl.conf** — файл конфигурации SSL-сертификата;
- **private.key** — файл приватного ключа SSL-сертификата;
- **request.csr** — файл запроса на выпуск SSL-сертификата;
- **rootCA.crt** — файл корневого сертификата от УЦ;
- **rootCA.key** — файл ключа корневого сертификата от УЦ;
- **domain.crt** — файл доверенного SSL-сертификата.

Чтобы добавить доверенный SSL-сертификат, выполните на управляющем узле кластера Kubernetes следующие действия:

1. Создайте файл **ssl.conf** с описанием конфигурации SSL-сертификата.
2. Сгенерируйте файлы приватного ключа **private.key** и запроса на выпуск сертификата **request.csr**:

```
openssl req -new -config ssl.conf -keyout private.key -out request.csr
```

3. Если у вас есть корневой сертификат **rootCA.crt** от УЦ и его ключ **rootCA.key**, подпишите с его помощью запрос **request.csr**:

```
openssl x509 -req -CA rootCA.crt -CAkey rootCA.key -in request.csr -out domain.crt
-days 365 -CAcreateserial
```

Если в конфигурации SSL-сертификата указано поле `subjectAltName`, то в конец команды необходимо добавить `-extfile ssl.conf -extensions req_ext`:

```
openssl x509 -req -CA rootCA.crt -CAkey rootCA.key -in request.csr
-out domain.crt -days 365 -CAcreateserial -extfile ssl.conf
-extensions req_ext
```

В результате будет создан доверенный сертификат **domain.crt**.

Если у вас нет корневого сертификата, передайте запрос **request.csr** удостоверяющему центру для создания доверенного сертификата.

4. Создайте секрет в кластере Kubernetes:

```
kubectl create secret tls <secret_name> --namespace <namespace> --key private.key
--cert domain.crt
```

Здесь:

- `<secret_name>` — имя секрета.
- `<namespace>` — пространство имен, в котором установлена система.

5. Внесите изменения в Ingress кластера Kubernetes.

Добавление самоподписанного SSL-сертификата

Самоподписанный SSL-сертификат — это сертификат, подписанный самим лицом, которое ответственно за ресурс. Самоподписанный сертификат не имеет статус доверенного в браузере пользователя до тех пор, пока не будет добавлен в список доверенных на компьютере пользователя вручную. Способ добавления самоподписанного сертификата в список доверенных отличается в зависимости от используемой ОС.

В представленной инструкции в качестве примера используются файлы со следующими названиями:

- **ssl.conf** — файл конфигурации SSL-сертификата;
- **private.key** — файл приватного ключа SSL-сертификата;
- **request.csr** — файл запроса на выпуск SSL-сертификата;
- **domain.crt** — файл самоподписанного SSL-сертификата.

Чтобы добавить самоподписанный SSL-сертификат, выполните на управляющем узле кластера Kubernetes следующие действия:

1. Создайте файл **ssl.conf** с описанием конфигурации SSL-сертификата.
2. Сгенерируйте файлы приватного ключа **private.key** и запроса на выпуск сертификата **request.csr**:

```
openssl req -new -config ssl.conf -keyout private.key -out request.csr
```

3. Создайте самоподписанный сертификат **domain.crt**:

```
openssl x509 -signkey private.key -in request.csr -req -days 365 -out domain.crt
```

Если в конфигурации SSL-сертификата указано поле `subjectAltName`, то в конец команды необходимо добавить `-extfile ssl.conf -extensions req_ext`:

```
openssl x509 -signkey private.key -in request.csr -req -days 365
-out domain.crt -extfile ssl.conf -extensions req_ext
```

4. Создайте секрет в кластере Kubernetes:

```
kubectl create secret tls <secret_name> --namespace <namespace> --key private.key
--cert domain.crt
```

Здесь:

- `<secret_name>` — имя секрета.
- `<namespace>` — пространство имен, в котором установлена система.

5. Внесите изменения в Ingress кластера Kubernetes.

Конфигурация SSL-сертификата

Конфигурация SSL-сертификата содержит набор обязательных и опциональных полей.

Пример 8. Пример конфигурации SSL-сертификата

```
# Полное доменное имя ресурса системы.
FQDN = <FQDN-адрес узла>

# Название организации.
ORGNAME = <название организации>

# (опционально) Альтернативные DNS-имена и IP-адреса, которые должны быть связаны с
сертификатом.
ALTNAMES = DNS:<DNS-имя 1>,DNS:<DNS-имя 2>,IP:<IP-адрес 1>,IP:<IP-адрес 2>

# Настройки для создания запроса сертификата CSR. Изменения не требуются.
[ req ]
default_bits = 2048
default_md = sha256
prompt = no
encrypt_key = no
distinguished_name = dn
req_extensions = req_ext
```

```
# Настройки уникального имени для сертификата, которые идентифицируют владельца
сертификата.
[ dn ]
C = CH
O = $ORGNAME
CN = $FQDN

# (опционально) Расширения, которые будут добавлены в запрос на сертификат.
[ req_ext ]
subjectAltName = $ALTNAMES
```

Общие поля

Общие поля задаются без привязки к какому-либо разделу полей.

Поле	Описание	Обязательное поле
`FQDN`	Полное доменное имя ресурса, на котором расположена система.	да
`ORGNAME`	Название организации, ответственной за ресурс.	да
`ALTNAMES`	Альтернативные IP-адреса и DNS-имена, которые должны быть связаны с сертификатом. Поле указывается при задании `subjectAltName = $ALTNAMES` в блоке `req_ext`.	нет

Блок полей `req`

В этом блоке задаются настройки для создания запроса сертификата CSR.

Рекомендуется оставлять этот блок без изменений.

Поле	Описание	Обязательное поле
`default_bits`	Указывает длину ключа сертификата в битах. Рекомендуется устанавливать длину 2048 бит.	да
`default_md`	Определяет алгоритм хеширования для подписания сертификата. Рекомендуется использовать алгоритм SHA-256.	да
`prompt`	Определяет, нужно ли запрашивать ввод данных для полей в блоках `dn` и `req_ext`. Рекомендуется устанавливать значение `no`. В этом случае ввод данных в указанных блоках не запрашивается. Вместо этого используются предопределенные значения в полях `distinguisged_name` и `req_extensions`.	да
`encrypt_key`	Определяет, нужно ли использовать ключ шифрования. Рекомендуется устанавливать значение `no`. В этом случае ключи шифрования не применяются.	да

Поле	Описание	Обязательное поле
`distinguished_name`	Указывает на блок полей, где задаются поля X.509. Рекомендуется указывать в поле ссылку на блок `dn`.	да
`req_extensions`	Указывает на блок полей, где задаются расширения. Рекомендуется указывать в поле ссылку на блок `req_ext`.	нет

Блок полей `dn`

Этот блок задает отличительное имя (Distinguished Name), идентифицирующее владельца сертификата.

Поле	Описание	Обязательное поле
`C`	Страна, в которой находится организация.	да
`O`	Название организации, ответственной за ресурс. Рекомендуется использовать значение поля `ORGNAME`.	да
`CN`	Общее имя (Common Name) ресурса, на которое будет выписан сертификат. Рекомендуется использовать значение поля `FQDN`.	да

Блок полей `req_ext`

Этот блок определяет расширения, которые будут добавлены в запрос на сертификат.

Поле	Описание	Обязательное поле
`subjectAltName`	Позволяет сертификату поддерживать несколько доменных имен и IP-адресов. Должен ссылаться на поле `ALTNAMES`.	нет

Изменение Ingress

Для изменения Ingress потребуется список зарезервированных доменов для системы. Рекомендуется подготовить его заранее.

Изменить Ingress можно одним из следующих способов:

- **Изменение Ingress вручную**:
  a. Откройте Ingress в режиме редактирования:

  ```
  kubectl edit ingress evo.core.host-app-host-app
  ```

  b. Добавьте в поле `spec` перед `rules` следующие строки:

```
tls:
  - <hosts>:
      - <доменное имя 1>
      - <доменное имя 2>
    <secret_name>: secret-name
```

Здесь:

- `<hosts>` — доменные имена, зарезервированные для системы.
- `<secret_name>` — имя секрета, созданного на шаге 4.

- **Изменение Ingress через helm-чарты**:

a. Создайте файл **values.yaml** со следующим содержанием:

```
modules:
  evo.core:
    host-app:
      ingress:
        ingresses:
          host-app:
            tls:
              - <hosts>:
                  - <доменное имя 1>
                  - <доменное имя 2>
                <secret_name>: secret-name
```

Здесь:

- `<hosts>` — доменные имена, зарезервированные для системы.
- `<secret_name>` — имя секрета, созданного на шаге 4.

b. Обновите сервис evo.core.host-app:

```
helm upgrade -n <namespace> evo.core.host-app /opt/r-vision/common/helm/r-huc
--reuse-values --values values.yaml
```

Здесь:

- `<namespace>` — пространство имен, в котором установлена система.

Добавление самоподписанного сертификата в список доверенных

Самоподписанный SSL-сертификат — это сертификат, подписанный самим лицом, которое ответственно за ресурс.

По умолчанию самоподписанный сертификат не имеет статус доверенного в браузере пользователя. Чтобы самоподписанный сертификат считался доверенным, его необходимо добавить в список

доверенных на компьютере пользователя.

Способ добавления самоподписанного сертификата в список доверенных отличается в зависимости от используемой ОС:

- Добавление сертификата в список доверенных в Windows
- Добавление сертификата в список доверенных в Linux
- Добавление сертификата в список доверенных в macOS

Добавление сертификата в список доверенных в Windows

Чтобы добавить самоподписанный сертификат в доверенные в Windows:

1. Откройте диспетчер сертификатов. Для этого выполните следующие действия:
   a. Нажмите сочетание клавиш WIN+R. Откроется окно **Выполнить**.
   b. Введите в окне `certlm.msc` и нажмите **ОК**.
2. Перейдите в раздел **Доверенные корневые центры сертификации** → **Сертификаты** с помощью дерева разделов в диспетчере сертификатов.
3. Выполните команду **Все задачи** → **Импорт** в контекстном меню раздела **Доверенные корневые центры сертификации** → **Сертификаты** в дереве разделов. Откроется мастер импорта сертификатов.
4. Нажмите **Далее** на стартовой странице мастера.
5. Выберите файл самоподписанного сертификата на странице выбора сертификата и нажмите **Далее**.
6. Удостоверьтесь, что на странице выбора хранилища выбран вариант **Поместить все сертификаты в следующее хранилище** и выбрано хранилище **Доверенные корневые центры сертификации**. Нажмите **Далее**.
7. Проверьте корректность указанных параметров на странице завершения импорта и нажмите **Готово**. Сертификат будет добавлен в список доверенных.

Добавление сертификата в список доверенных в Linux

Чтобы добавить самоподписанный сертификат в доверенные в Linux:

- Для Debian-подобных дистрибутивов:
  1. Перейдите в каталог, в котором содержится файл самоподписанного сертификата.
  2. Скопируйте файл самоподписанного сертификата в каталог `/usr/local/share/ca-certificates/`:

     ```
     sudo cp domain.crt /usr/local/share/ca-certificates/domain.crt
     ```

     Здесь `domain.crt` — файл самоподписанного сертификата.
  3. Выполните команду:

```
sudo update-ca-certificates
```

Сертификат будет добавлен в список доверенных.

- Для RedHat-подобных дистрибутивов: действия по настройке доверенных сертификатов могут отличаться в зависимости от дистрибутива.

Точный алгоритм настройки и требуемые команды необходимо смотреть в официальной документации для конкретного дистрибутива.

Основной алгоритм действий:

1. Добавьте файл самоподписанного сертификата в каталог `/etc/pki/ca-trust/source/anchors/`.
2. Выполните команду:

```
update-ca-trust extract
```

Сертификат будет добавлен в список доверенных.

Добавление сертификата в список доверенных в macOS

Чтобы добавить самоподписанный сертификат в доверенные в macOS:

1. Откройте приложение **Связка ключей**.
2. Откройте приложение **Finder** и найдите в нем файл самоподписанного сертификата. Перетащите найденный файл в окно приложения **Связка ключей**.
3. Перейдите на вкладку **Сертификаты** в приложении **Связка ключей**.
4. Дважды нажмите на добавленный самоподписанный сертификат. Откроется окно сведений о сертификате.
5. Раскройте группу параметров **Доверие** и для каждого параметра выберите вариант **Всегда доверять**. Сертификат будет добавлен в список доверенных.

Работа с кластером

Данный раздел предназначен для администраторов и разработчиков, ответственных за управление и мониторинг кластера Kubernetes, в котором развернута система. В нем описаны инструменты и методы контроля состояния кластера и управления его компонентами.

Содержание раздела:

- Подключение к кластеру
- Мониторинг кластера
  - Состояние кластера
  - Создание пользовательских дашбордов в Grafana

    - Анализ работоспособности сервисов кластера
    - Анализ работоспособности контрольной плоскости кластера
    - Анализ состояния сателлита
    - Просмотр журнала событий кластера
    - Просмотр состояния подов кластера
    - Список подов сервисов
    - Настройка оповещений
        - Настройка оповещений в Telegram
        - Настройка оповещений по электронной почте
    - Работа с метриками Prometheus
- Управление кластером
    - Управление узлами кластера
    - Работа с подом collector-manager
    - Увеличение размера тома для подов
    - Резервное копирование и восстановление PostgreSQL
    - Резервное копирование хранилища etcd
    - Получение идентификатора сервиса в NATS Service Discovery
    - Работа с объектным хранилищем NATS Object Store
- Работа с паролями в кластере
- Изменение доменного имени установленного экземпляра системы

Подключение к кластеру

В данном разделе описано, как подключить кластер Kubernetes к Lens — рабочему столу для управления кластерами.

О Lens

Lens — это рабочий стол для управления кластерами Kubernetes, предоставляющий интерфейс для мониторинга и администрирования.

Установка Lens

1. Скачайте установщик Lens для вашей операционной системы с официального сайта и выполните установку на ваше локальное устройство, с которого вы планируете подключаться к кластеру.
2. Запустите установленное приложение и войдите в свою учетную запись **Lens ID**. Если у вас уже есть **Lens ID**, то кликните **Login with Lens ID**, введите свои реквизиты и выполните вход.

После успешной авторизации откроется рабочее меню приложения Lens.

Подключение к кластеру

Указание kubeconfig файлов

Lens автоматически обнаруживает кластеры из указанных kubeconfig файлов. Доступные кластеры можно найти в разделе **Catalog**.

Чтобы подключиться к кластеру с помощью Lens:

1. Перейдите в раздел **Catalog** → **Clusters**.
2. Нажмите на кнопку **Add Cluster** в правом нижнем углу.
3. Добавьте файл (или файлы) конфигурации кластера или папку с файлами. Выберите один из следующих вариантов:
   - **Sync kubeconfig file(s)** — путь к конкретным kubeconfig файлам.
   - **Sync kubeconfig folder(s)** — путь к папкам, содержащим kubeconfig файлы.

Файл конфигурации кластера (обычно `kubeconfig`) содержит информацию о кластере и учетные данные для доступа к нему. Вы можете получить этот файл у администратора кластера или сгенерировать его самостоятельно с помощью командной строки Kubernetes (kubectl).

4. При успешном добавлении кластер отобразится в списке.
5. Выберите строку с нужным кластером в списке для подключения. Откроется окно **Cluster Overview** с подробной информацией о выбранном кластере.

Ручное добавление кластера

Lens поддерживает ручное добавление кластеров через меню **Add from Kubeconfig**. В этом меню можно указать отдельный kubeconfig файл, отличный от указанного в меню настроек **Preferences**.

Kubeconfig файл, добавленный через меню **Add from Kubeconfig**, не объединяется с kubeconfig файлом, указанным в меню **Preferences**.

Для ручного добавления кластера:

1. Перейдите в раздел **Catalog** → **Clusters**.
2. Наведите курсор на кнопку **Add Cluster**.
3. Выберите **Add from kubeconfig** и вставьте пользовательский kubeconfig файл в соответствующее поле ввода.

Просмотр текущей конфигурации кластера

Для просмотра текущей конфигурации кластера:

1. В окне кластера нажмите кнопку **New tab** и выберите **Terminal session**.
2. Во вкладке **Terminal** выполните следующую команду:

```
kubectl config view --minify --raw
```

При подключении к кластеру убедитесь, что у вас есть действительный и рабочий kubeconfig файл для кластера.

Удаление кластера из Lens

Вы можете удалить кластер из списка кластеров как в меню **Catalog**, так и в окне кластера. Удаление кластера из Lens не влияет на сам кластер или его конфигурацию.

Мониторинг кластера

В данном разделе представлена информация о методах и инструментах мониторинга состояния кластера Kubernetes, используемых для обеспечения стабильности и высокой доступности системных ресурсов.

Содержание раздела:

- Состояние кластера
- Создание пользовательских дашбордов в Grafana
- Анализ работоспособности сервисов кластера
- Анализ работоспособности контрольной плоскости кластера
- Анализ состояния сателлита
- Просмотр журнала событий кластера
- Просмотр состояния подов кластера
- Список подов сервисов
- Настройка оповещений
  - Настройка оповещений в Telegram
  - Настройка оповещений по электронной почте
- Работа с метриками Prometheus

Состояние кластера

Наблюдение за состоянием кластера Kubernetes осуществляется в разделе **Инструменты** → **Мониторинг**. В нем предоставлена подробная информация о контейнерах, узлах, деплоях и других компонентах кластера.

О мониторинге состояния кластера

Мониторинг состояния кластера Kubernetes в системе осуществляется с помощью решения Grafana, предоставляющего визуализацию собранных метрик. Раздел **Инструменты** → **Мониторинг** в веб-интерфейсе системы отображает полную информацию о состоянии кластера, контейнеров, узлов и других компонентов.

При переходе в раздел **Инструменты** → **Мониторинг** открывается портал Grafana, развернутый во

время установки системы, который автоматически обновляется и предоставляет актуальные данные о состоянии кластера. Раздел отображает различные метрики и диаграммы, позволяя пользователям следить за производительностью и состоянием кластера в реальном времени.

Основные возможности раздела включают:

- Отслеживание состояния контейнеров, включая запущенные, ожидающие и завершенные контейнеры, а также количество перезапусков контейнеров за последние 30 минут.
- Мониторинг состояния узлов, отображающий количество доступных и недоступных узлов, а также узлов, не готовых к работе.
- Анализ запросов и доступных ресурсов в кластере, включая поды, CPU и оперативную память.
- Отображение состояния горизонтальных подов (HPA), заданий (Jobs) и постоянных объемов (PVC).
- Наблюдение за состоянием `statefulset` реплик и перезапусками подов.

Разделы мониторинга

В разделе **Инструменты → Мониторинг** представлены различные вкладки, каждая из которых отображает определенные аспекты работы кластера Kubernetes:

- **Cluster** — отображает общие метрики по кластеру, включая запросы и доступные ресурсы, такие как поды, CPU и оперативная память.
- **Node** — предоставляет информацию о состоянии узлов кластера, включая их количество, доступность и готовность к работе.
- **Deployments** — показывает состояние деплоев в кластере, в том числе количество реплик и их состояние.
- **Statefulset** — отображает информацию о `statefulset` репликах, включая их актуальное состояние и статус выселения.
- **Pods** — демонстрирует состояние подов, включая их количество, статусы выполнения и перезапуска.
- **Containers** — отображает состояние контейнеров в кластере.
- **HPA** — информирует о горизонтальном масштабировании подов (HPA), отображая текущее состояние и пределы масштабирования.
- **Jobs** — отображает статус заданий в кластере, включая количество успешно выполненных, активных и проваленных заданий.
- **PVC** — предоставляет данные по постоянным томам (PVC), указывая на их связывание и доступность.

Cluster

Вкладка **Cluster** содержит метрики по запросам и доступным ресурсам в кластере:

- **Cluster Pod Requested** — количество запрашиваемых подов.
- **Cluster CPU Requested** — количество запрашиваемых ядер процессора.
- **Cluster Memory Requested** — количество запрашиваемой оперативной памяти.

- **Cluster Pod Capacity** — доступная и используемая емкость подов в кластере.
- **Cluster CPU Capacity** — доступная и используемая емкость CPU в кластере.
- **Cluster Mem Capacity** — доступная и используемая емкость оперативной памяти в кластере.

Node

Вкладка **Node** предоставляет информацию о состоянии узлов:

- **Number of Nodes** — общее количество узлов в кластере.
- **Nodes Unavailable** — количество недоступных узлов.
- **Node NotReady** — узлы, не готовые к работе.
- **Node Pressure** — давление на узлы.

Deployments

Вкладка **Deployments** показывает информацию о деплоях в кластере, включая:

- **Deployment Replicas - Up To Date** — количество реплик деплоя, которые соответствуют ожидаемому состоянию.
- **Deployment Replicas** — текущее состояние реплик деплоя.
- **Deployment Replicas - Updated** — количество недавно обновленных реплик.
- **Deployment Replicas - Unavailable** — количество недоступных реплик.

Statefulset

Вкладка **Statefulset** показывает состояние `statefulset` реплик:

- **Statefulset Replicas - Up To Date** — количество актуальных реплик `statefulset`.

Pods

Вкладка **Pods** отображает состояние подов:

- **Pods restart in 30m** — количество перезапусков подов за последние 30 минут.
- **Pods Running** — количество работающих подов.
- **Pods Succeeded** — количество успешно завершенных подов.
- **Pods Failed** — количество проваленных подов.
- **Pods Pending** — количество подов, ожидающих запуска.
- **Pods Unknown** — количество подов в неизвестном состоянии.

Containers

Вкладка **Containers** отображает состояние контейнеров в кластере:

- **Containers Running** — количество запущенных контейнеров.
- **Containers Waiting** — количество контейнеров, ожидающих запуска.

- **Containers Terminated** — количество завершенных контейнеров.
- **Containers Restarts (Last 30 Minutes)** — количество перезапусков контейнеров за последние 30 минут.
- **CPU Cores Requested by Containers** — количество запрашиваемых контейнерами ядер процессора.
- **Memory Requested by Containers** — количество запрашиваемой контейнерами оперативной памяти.

HPA

Вкладка **HPA** отображает горизонтальные поды:

- **total-hpa-current** — текущее состояние всех HPA.
- **current==max** — максимальное текущее значение.
- **current==min** — минимальное текущее значение.

Jobs

Вкладка **Jobs** показывает состояние заданий в кластере:

- **Jobs Succeeded** — количество успешно завершенных заданий.
- **Jobs Active** — количество активных заданий.
- **Jobs Failed** — количество проваленных заданий.
- **Job last success time from now** — время последнего успешного выполнения задания.

PVC

Вкладка **PVC** отображает информацию о постоянных объемах:

- **pvc is bound** — количество связанных PVC.
- **pvc is not bound** — количество несвязанных PVC.

Создание пользовательских дашбордов в Grafana

Вы можете создавать дашборды в Grafana для мониторинга состояния кластера. Чтобы создать новый дашборд:

1. Войдите в веб-интерфейс Grafana.
2. В разделе **Dashboards** нажмите на кнопку **New** и выберите **New dashboard**.
3. Чтобы перейти к созданию первой панели дашборда, на следующем экране нажмите **Add visualization**.
4. Чтобы добавить новую панель на существующий дашборд, нажмите **Add** на верхней панели инструментов, затем выберите **Visualization** в выпадающем меню.

Примеры создания панелей с дополнительными метриками состояния кластера приведены ниже. Подробная информация доступна в официальной документации Grafana.

Примеры панелей пользовательских метрик в Grafana

Создание графика с метриками из Prometheus

В Grafana можно выводить графики метрик, собранных системой Prometheus. Ниже приведен пример построения графика запросов подов кластера на ресурсы процессора и памяти.

Чтобы добавить на дашборд график метрик из Prometheus, выполните следующие действия:

1. Создайте новую панель на дашборде.
2. Выберите источник данных **Prometheus**. Это можно сделать:
   a. при создании панели в окне **Select data source**;
   b. в нижней части экрана, в выпадающем списке **Data source**.
3. Выберите тип визуализации **Time series** из выпадающего списка в правом верхнем углу экрана.
4. На панели настроек запроса (по умолчанию **A**) выберите метрику из выпадающего списка **Metric**:
   a. Для CPU: `cluster:namespace:pod_cpu:active:kube_pod_container_resource_requests`.
   b. Для оперативной памяти: `cluster:namespace:pod_memory:active:kube_pod_container_resource_requests`.

По умолчанию метрики оперативной памяти отображаются в байтах. Можно преобразовать их в более информативные единицы измерения, например, в мегабайты. Для этого используйте деление на число-делитель:

- На панели настроек запроса нажмите на кнопку **Operations**.
- Выберите **Binary operations** → **Divide by scalar**.
- Укажите делитель в поле **Value**. Например, `1048576` для перевода в мегабайты.

Также в списке **Binary operations** доступны другие арифметические операции над значениями графика.

5. Для фильтрации подов по различным критериям, таким как принадлежность к пространству имен или узлу, используйте меню **Label filters** справа от выпадающего списка **Metric**.
6. Чтобы настроить подписи в легенде графика:
   - Откройте выпадающее меню **Options** в нижней части панели настроек запроса.
   - В выпадающем списке **Legend** выберите опцию **Custom**.
   - Укажите текст подписи, например, `{{pod}} CPU request`, где ключевое слово `{{pod}}` будет заменено на имя пода.

В правой панели редактора доступны дополнительные настройки графика, такие как:

- Изменение названия графика в поле **Title**.
- Настройка сортировки значений в легенде графика. Чтобы ее применить:
  - В разделе **Legend** установите режим отображения (**Mode**) на **Table**.

- В поле **Values** выберите показатели для отображения в таблице легенды. Например, среднее значение и максимум за выбранный временной период.
- Выбранные показатели отобразятся в таблице легенды. Для сортировки графиков по ним нажмите на заголовки соответствующих столбцов.

Вывод логов Loki

Grafana позволяет отображать логи, собранные системой Loki. Ниже приведен пример вывода логов шардов ClickHouse.

Чтобы добавить на дашборд панель логов, собранных Loki, выполните следующие действия:

1. Создайте новую панель на дашборде.
2. Выберите источник данных **DS_LOKI**. Это можно сделать:
   a. при создании панели в окне **Select data source**;
   b. в нижней части экрана, в выпадающем списке **Data source**.
3. Выберите тип визуализации **Logs** из меню в правом верхнем углу экрана.
4. На панели настроек запроса (по умолчанию **A**) настройте фильтрацию логов, используя меню **Label filters**. Выберите метки объектов кластера, с которых должны быть получены логи: например, пространство имен, контейнер или под.

   Например, для отображения логов шардов ClickHouse в меню **Label filters** задайте:

   - метку `pod`;
   - оператор `=~`;
   - имена подов шардов ClickHouse, например, `clickhouse-shard0-0`, `clickhouse-shard1-0`.

   Строки логов можно фильтровать по заданным условиям. Для этого:

   - На панели настроек запроса нажмите на кнопку **Operations**.
   - Выберите **Line filters** в выпадающем списке.
   - Укажите тип фильтра. Доступны фильтры с учетом регистра и регистронезависимые.

   Например, чтобы отобразить только сообщения, содержащие текст **error** без учета регистра:

   - Выберите тип фильтра **Line contains case insensitive**.
   - В поле **Text to find** введите `error`.

5. В меню **Options** в нижней части панели настроек запроса можно задать лимит числа строк логов, выводимых на панель, с помощью поля **Line limit**.

В правой панели редактора доступны дополнительные настройки графика, такие как:

- Изменение названия графика в поле **Title**.

- Настройка отображения логов в разделе **Logs**, например:
  - Добавление метки времени в часовом поясе пользователя с помощью флажка **Time**.
  - Включение переноса строк логов с помощью флажка **Wrap lines**.

Сохранение дашборда

После создания панелей сохраните дашборд, нажав на кнопку **Save dashboard** ( ) в верхней панели инструментов. Сохраненный дашборд будет доступен в разделе **Dashboards** веб-интерфейса Grafana.

Анализ работоспособности сервисов кластера

Для обеспечения работы кластера используются следующие сервисы:

- Dkron;
- ClickHouse;
- NATS Key/Value & Service Discovery

В данном разделе описаны способы проверки работоспособности этих сервисов.

Состояние NATS можно проверить только с помощью командной строки, а всех остальных сервисов — с помощью как командной строки, так и функции проброса портов рабочего стола Lens.

Проверка состояния подов сервисов с помощью командной строки

Просмотреть состояние, логи подов и подробную информацию о них можно с помощью команд, описанных в разделе **Просмотр состояния подов кластера**.

Проверка состояния подов сервисов с помощью проброса портов

Часть сервисов, используемых кластером, имеет панели управления и методы API, с помощью которых можно получить подробную информацию об их состоянии. Подключиться к ним можно с помощью функции проброса портов рабочего стола Lens.

Проверка состояния Dkron

Чтобы получить доступ к панели управления распределенного планировщика задач Dkron:

1. Подключитесь к кластеру с помощью Lens.
2. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Pods**.
3. Выберите пространство имен, в котором установлен кластер, в выпадающем списке **Select Namespace** в правой части экрана. Отобразится таблица подов кластера.
4. В поле поиска **Search Pods...** введите **dkron**. Отобразится под, на котором размещен экземпляр Dkron.
5. Нажмите на под левой кнопкой мыши. В правой части экрана откроется панель с подробной информацией о поде.
6. В разделе **Ports** панели информации найдите ссылку на сервис **http**: например, `http: 8080/TCP`.

7. Нажмите на ссылку. В окне браузера откроется страница панели управления Dkron.

На панели управления Dkron отображается статистика запланированных, выполненных и проваленных заданий, а также текущее состояние узлов Dkron.

Проверка состояния ClickHouse

Чтобы получить доступ к HTTP API СУБД ClickHouse:

1. Подключитесь к кластеру с помощью Lens.
2. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Pods**.
3. Выберите пространство имен, в котором установлен кластер, в выпадающем списке **Select Namespace** в правой части экрана. Отобразится таблица подов кластера.
4. В поле поиска **Search Pods…** введите **clickhouse**. Отобразятся поды, на которых размещены шарды ClickHouse.
5. Нажмите на любой под левой кнопкой мыши. В правой части экрана откроется панель с подробной информацией о поде.
6. В разделе **Ports** панели информации найдите ссылку на сервис **http**: например, `http: 8123/TCP`.
7. Нажмите на ссылку. В окне браузера откроется страница, на которой отобразится сообщение о состоянии ClickHouse.

При нормальной работе ClickHouse, по умолчанию на странице отображается сообщение `Ok`. Сообщение может быть изменено в конфигурации ClickHouse.

Чтобы открыть графический клиент для выполнения запросов к СУБД, добавьте `/play` к URL. Например, `http://localhost:61399/play`.

Для использования клиента введите логин и пароль администратора ClickHouse в поля в правом верхнем углу экрана.

Логин и пароль можно найти в разделе **Environment** информации о поде в Lens. Они указаны в переменных `CLICKHOUSE_ADMIN_USER` и `CLICKHOUSE_ADMIN_PASSWORD`

Анализ работоспособности контрольной плоскости кластера

В случае возникновения проблем в работе кластера, в особенности после перезапуска, следует проверить состояние его плоскости управления (control plane). Плоскость управления кластера включает следующие компоненты:

- `kube-apiserver` — компонент, который представляет API Kubernetes;
- `etcd` — хранилище конфигураций и состояний ресурсов кластера Kubernetes;
- `kube-scheduler` — компонент, который отслеживает созданные поды без привязанного узла и выбирает узел, на котором они должны работать;
- `kube-controller-manager` — диспетчер контроллеров.

Чтобы проверить состояние плоскости управления, выполните следующие действия в командной строке:

1. Проверьте состояние компонентов плоскости управления с помощью команды:

```
kubectl get --raw='/readyz?verbose'
```

Все компоненты должны иметь состояние `ok`:

▼ *Вывод команды* `kubectl get --raw='/readyz?verbose'`, *если кластер функционирует нормально*

```
[+]ping ok
[+]log ok
[+]etcd ok
[+]etcd-readiness ok
[+]informer-sync ok
[+]poststarthook/start-kube-apiserver-admission-initializer ok
[+]poststarthook/generic-apiserver-start-informers ok
[+]poststarthook/priority-and-fairness-config-consumer ok
[+]poststarthook/priority-and-fairness-filter ok
[+]poststarthook/storage-object-count-tracker-hook ok
[+]poststarthook/start-apiextensions-informers ok
[+]poststarthook/start-apiextensions-controllers ok
[+]poststarthook/crd-informer-synced ok
[+]poststarthook/bootstrap-controller ok
[+]poststarthook/rbac/bootstrap-roles ok
[+]poststarthook/scheduling/bootstrap-system-priority-classes ok
[+]poststarthook/priority-and-fairness-config-producer ok
[+]poststarthook/start-cluster-authentication-info-controller ok
[+]poststarthook/start-kube-apiserver-identity-lease-controller ok
[+]poststarthook/start-kube-apiserver-identity-lease-garbage-collector ok
[+]poststarthook/start-legacy-token-tracking-controller ok
[+]poststarthook/aggregator-reload-proxy-client-cert ok
[+]poststarthook/start-kube-aggregator-informers ok
[+]poststarthook/apiservice-registration-controller ok
[+]poststarthook/apiservice-status-available-controller ok
[+]poststarthook/kube-apiserver-autoregistration ok
[+]autoregister-completion ok
[+]poststarthook/apiservice-openapi-controller ok
[+]poststarthook/apiservice-openapiv3-controller ok
[+]shutdown ok
readyz check passed
```

Если состояние компонента не соответствует значению `ok`, изучите сообщение об ошибке для этого компонента. Например, оно может быть следующим: `[-]etcd failed: reason withheld`.

Если по сообщению об ошибке не удалось установить ее причину, проверьте состояние подов компонентов, выполнив шаг 2.

2. Для проверки состояния подов компонентов выполните следующую команду:

```
kubectl get pods -n kube-system -o wide | grep <component>
```

Здесь:

- `<component>` — имя компонента из списка выше.

  Если компонент работает корректно, в выводе команды под должен иметь статус `Running` (третий столбец). Если статус пода отличается от `Running`, проанализируйте логи пода (шаг 3).

Если команда `kubectl get pods -n kube-system -o wide | grep etcd` не возвращает результатов, это означает, что компонент `etcd` установлен как сервис. Для проверки его состояния используйте следующую команду:

```
systemctl status etcd.service
```

▼ *Пример вывода состояния компонента* `etcd`

```
● etcd.service - etcd
     Loaded: loaded (/etc/systemd/system/etcd.service;
enabled; vendor preset: disabled)
     Active: active (running) since Mon 2024-09-16 14:27:48
MSK; 2 months 13 days ago
   Main PID: 695 (etcd)
      Tasks: 12 (limit: 4652)
     Memory: 403.8M
        CPU: 1d 5h 58min 7.285s
     CGroup: /system.slice/etcd.service
             └─ 695 /usr/local/bin/etcd

Nov 29 14:07:35 master01-inst.test.coe.rvision.local
etcd[695]: {"level":"info","ts":"2024-11-
29T14:07:35.980+0300","caller":"mvcc/hash.go:137","msg":"stori
ng new hash","hash":669749590,"revision":42723972,"compact-
revision":42723201}
Nov 29 14:12:35 master01-inst.test.coe.rvision.local
etcd[695]: {"level":"info","ts":"2024-11-
29T14:12:35.935+0300","caller":"mvcc/index.go:214","msg":"comp
act tree index","revision":42724748}
...
```

Если сервис `etcd` работает корректно, его статус (`Active`) отображается как `active (running)`. В противном случае обратитесь в службу поддержки, приложив вывод команды `systemctl status etcd.service`.

3. Если статус пода сервиса отличается от `Running`, выполните команду для получения его логов:

```
kubectl logs -n kube-system <pod_name> --all-containers=true
```

Здесь:

- `<pod_name>` — имя пода, полученное на шаге 2.

При большом количестве событий в логе можно сузить вывод, перенаправив результаты команды `kubectl logs` в утилиты для обработки текста. Например, чтобы отобразить только последние несколько событий:

Пример 9. Пример вывода 20 последних событий в журнале с помощью команды `tail`

```
kubectl logs -n kube-system kube-scheduler-master.rvision.local
--all-containers=true | tail -n 20
```

Если логи пода недоступны, можно просмотреть таблицу последних событий (`Events`), связанных с подом. Для этого выполните команду:

```
kubectl -n kube-system describe pod <pod_name>
```

Здесь:

- `<pod_name>` — имя пода, полученное на шаге 2.

4. Если анализ логов или таблицы `Events` не выявил причину неисправности кластера, обратитесь в службу поддержки и приложите вывод команд, выполненных на шагах 1—3.

Анализ состояния сателлита

Если в системе установлены сателлиты, но установленные на них коллекторы или сервисы не работают, можно проанализировать состояние подов кластера сателлита. Для этого выполните следующие действия:

1. Подключитесь к терминалу управляющего узла кластера сателлита.
2. Выполните следующую команду:

```
kubectl -n <namespace> get pods
```

Здесь:

- `<namespace>` — имя пространства имен сателлита.

Если сателлит работает корректно, все его поды имеют статус `Ready`. Если есть поды, статус которых отличается от `Ready`, или все поды имеют статус `Ready`, но сателлит работает некорректно, обратитесь в службу поддержки по адресу support@rvision.ru.

Просмотр журнала событий кластера

С помощью командной строки

Чтобы просмотреть журнал событий кластера, выполните следующую команду:

```
kubectl -n <namespace> events
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.

▼ *Пример журнала событий кластера*

```
LAST SEEN                 TYPE      REASON                    OBJECT
MESSAGE
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-14277d5d-b2ac-4c7f-a14e-261aa5345751-correlator1
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-464a69cb-93cf-4111-9907-4445237a24e5-correlator0
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-464a69cb-93cf-4111-9907-4445237a24e5-correlator1
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-464a69cb-93cf-4111-9907-4445237a24e5-correlator2
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-53027b15-cb78-4019-aacc-2d09232c30ff-correlator0
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-53027b15-cb78-4019-aacc-2d09232c30ff-correlator2
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-7083cdce-3ecf-4968-93ae-e4b7058edc68
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-14277d5d-b2ac-4c7f-a14e-261aa5345751-correlator0
waiting for first consumer to be created before binding
2m53s (x115887 over 20d)   Normal    WaitForFirstConsumer
PersistentVolumeClaim/collector-a4a12d20-c14e-48aa-9c4e-4977f9a2e6cb-correlator0
waiting for first consumer to be created before binding
119s (x183 over 15h)       Warning   VolumeConditionAbnormal
PersistentVolumeClaim/collector-7766b6c7-8bfa-4aa4-b7c0-351687f737fa
Resource with issues on node(s): evo-worker02.k8s.dev.rvision.local
119s (x183 over 15h)       Warning   VolumeConditionAbnormal
PersistentVolumeClaim/data-clickhouse-shard0-0
Resource with issues on node(s): evo-worker02.k8s.dev.rvision.local
119s (x183 over 15h)       Warning   VolumeConditionAbnormal
PersistentVolumeClaim/dkron
```

```
Resource with issues on node(s): evo-worker02.k8s.dev.rvision.local
119s (x183 over 15h)       Warning   VolumeConditionAbnormal
PersistentVolumeClaim/collector-edd90ee5-2bc2-4c2a-9513-ace74b7a5d56
Resource with issues on node(s): evo-worker02.k8s.dev.rvision.local
119s (x183 over 15h)       Warning   VolumeConditionAbnormal
PersistentVolumeClaim/collector-8f80fa2a-63a2-4986-9447-80a7a74a93c6-correlator0
Resource with issues on node(s): evo-worker02.k8s.dev.rvision.local
...
```

При большом количестве событий в списке можно сузить вывод, передав результаты команды `kubectl -n <namespace> events` в утилиты обработки текста. Например, чтобы отобразить только последние несколько событий:

Пример 10. Вывод 20 последних событий в журнале с помощью команды `tail`

```
kubectl -n <namespace> events | tail -n 20
```

С помощью Lens

Чтобы просмотреть журнал событий кластера:

1. Подключитесь к кластеру с помощью Lens.
2. В выпадающем меню свойств кластера выберите пункт **Events**.
3. Выберите пространство имен, в котором установлен кластер, в выпадающем списке **Select Namespace** в правой части экрана. Отобразится таблица событий кластера.

Просмотр состояния подов кластера

Просмотр статуса и логов подов может понадобиться при решении проблем с системой. Инструкция по поиску подов, на которых расположены сервисы системы, описана в разделе Список подов сервисов.

Просмотр состояния подов в списке

Чтобы просмотреть состояние подов кластера, выполните следующую команду:

```
kubectl -n <namespace> get pods
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.

▼ *Пример списка подов кластера*

```
NAME                        READY   STATUS      RESTARTS         AGE
clickhouse-shard0-0         1/1     Running     1 (7h53m ago)    2d3h
clickhouse-shard1-0         1/1     Running     0                2d3h
....
```

```
nats-0                    3/3     Running     0               22d
nats-box-575              1/1     Running     0               22d
pgbouncer-846c            2/2     Running     1 (27d ago)     35d
postgresql-0              1/1     Running     0               64d
```

Состояние подов отображается в столбце `STATUS`. Возможные статусы:

- `Pending`: под принят кластером, но один или несколько контейнеров еще не установлены и не подготовлены к запуску. Этот статус включает время, в течение которого под ожидает планирования, а также загружаются по сети образы контейнеров.
- `Running`: под привязан к узлу, а также все его контейнеры созданы. По крайней мере один контейнер работает или находится в процессе запуска или перезапуска.
- `Succeeded`: все контейнеры пода успешно завершили работу и не будут перезапущены.
- `Failed`: все контейнеры пода завершили работу, и по крайней мере один контейнер завершил работу с ошибкой (с кодом, не равным 0, или был остановлен системой) и не будет автоматически перезапущен.
- `Unknown`: состояние пода не может быть получено. Такое состояние обычно возникает из-за ошибки связи с узлом, в котором под должен работать.

Также состояние подов кластера можно просмотреть на дашборде Grafana в разделе системы **Инструменты** → **Мониторинг**.

Просмотр подробной информации о поде

Чтобы просмотреть подробную информацию о поде, в том числе состояние, узел, на котором он находится, список его контейнеров и так далее, выполните следующую команду:

```
kubectl -n <namespace> describe pods/<pod_name>
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<pod_name>` — имя пода, информацию о котором требуется просмотреть.

▼ *Пример вывода команды* `kubectl describe`

```
Name:             test-pod
Namespace:        evo
Priority:         0
Service Account:  default
Node:             dev.evo.local/10.0.0.13
Start Time:       Wed, 30 Oct 2024 14:49:49 +0300
Labels:           app.kubernetes.io/component=nats
                  app.kubernetes.io/instance=nats
                  app.kubernetes.io/managed-by=Helm
                  app.kubernetes.io/name=nats
                  app.kubernetes.io/version=2.10.20
```

```
                  controller-revision-hash=nats-56bcf785dd
                  helm.sh/chart=nats-1.2.4
                  statefulset.kubernetes.io/pod-name=nats-0
Annotations:      checksum/config:
8b3d84c2a7387f68211b5f98f2d6d910a2c0491fcaef10d1187d3d1fac81dea9
                  cni.projectcalico.org/containerID:
d1f8a2f334f8f1c9a47c835ef92f6e95b8cf43c1e085982de2c396c972203e9f
                  cni.projectcalico.org/podIP: 10.0.0.191/32
                  cni.projectcalico.org/podIPs: 10.0.0.191/32
Status:           Running
IP:               10.0.0.191
IPs:
  IP:           10.0.0.191
Controlled By:  StatefulSet/nats
Containers:
  nats:
    Container ID:
containerd://b3638b859a891ae7b6dc897ec50aa234ff09698decf4eccb5adca0a0794f7e24
    Image:         sec/nats-server:2.10.20-scratch
    Image ID:      sec/nats-
server@sha256:d5508b97e054c298bba593e55ec35b7c5a164def88a4ccd9c456d4b598d1a674
    Ports:         4222/TCP, 8222/TCP
    Host Ports:    0/TCP, 0/TCP
    Args:
      --config
      /etc/nats-config/nats.conf
    State:          Running
      Started:      Wed, 30 Oct 2024 14:49:51 +0300
    Ready:          True
    Restart Count:  0
    Liveness:       http-get http://:monitor/healthz%3Fjs-enabled-only=true
delay=10s timeout=5s period=30s #success=1 #failure=3
    Readiness:      http-get http://:monitor/healthz%3Fjs-server-only=true delay=10s
timeout=5s period=10s #success=1 #failure=3
    Startup:        http-get http://:monitor/healthz delay=10s timeout=5s period=10s
#success=1 #failure=90
    Environment:
      POD_NAME:     nats-0 (v1:metadata.name)
      SERVER_NAME:  $(POD_NAME)
    Mounts:
      /etc/nats-config from config (rw)
      /var/run/nats from pid (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-4fwsp (ro)
Events:           <none>
```

Раздел `Events` в выводе команды `kubectl describe` отображает последние события, связанные с подом. Эти данные полезны для диагностики и устранения проблем в работе пода.

Подробная информация о команде `kubectl describe` доступна в документации

Kubernetes.

Просмотр логов пода

Логи пода очищаются при его остановке.

Чтобы просмотреть логи контейнеров, расположенных на поде, выполните команду `kubectl logs`.

Просмотр логов конкретного контейнера:

```
kubectl -n <namespace> logs <pod_name> -c <container_name>
```

Просмотр логов всех контейнеров на поде:

```
kubectl -n <namespace> logs <pod_name> --all-containers=true
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<pod_name>` — имя пода, логи которого требуется просмотреть.
- `<container_name>` — имя контейнера, логи которого требуется просмотреть.

Чтобы логи передавались в потоковом режиме, то есть обновлялись в реальном времени, укажите опцию `-f` команды `kubectl logs`:

```
kubectl -n <namespace> logs -f <pod_name> --all-containers=true
```

С подробной справкой о команде `kubectl logs` можно ознакомиться в документации Kubernetes.

Просмотр потребления подом ресурсов процессора и памяти

Чтобы просмотреть потребление подом ресурсов процессора и памяти, выполните следующую команду:

```
kubectl -n <namespace> top pod <pod_name>
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<pod_name>` — имя пода, информацию о котором требуется просмотреть.

▼ *Пример вывода команды* `kubectl top pod`

```
NAME       CPU(cores)   MEMORY(bytes)
```

```
test-pod   29m          109Mi
```

Чтобы наблюдать за изменением потребления ресурсов со временем, используйте команду выше как аргумент команды `watch`:

```
watch -n<seconds> kubectl -n <namespace> top pod <pod_name>
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<pod_name>` — имя пода, информацию о котором требуется просмотреть.
- `<seconds>` — число секунд между запусками команды, являющейся аргументом `watch`.

С подробной справкой о команде `kubectl top pod` можно ознакомиться в документации Kubernetes.

Список подов сервисов

Каждый сервис системы имеет уникальное имя, с помощью которого можно найти под кластера, на котором он расположен. В этом разделе приведен список сервисов системы, а также команды для получения полных имен подов, на которых они размещены.

Поды сервисов в пространстве имен кластера

Большая часть подов сервисов находится в пространстве имен кластера и доступна по селектору `app.kubernetes.io/name`.

Каждому поду присваивается случайный идентификатор, добавляемый к имени сервиса, для которого создается под.

Чтобы получить полное имя пода сервиса в пространстве имен кластера, выполните следующую команду:

```
kubectl -n <namespace> get pods -l app.kubernetes.io/name=<service_name>
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<service_name>` — имя сервиса.

▼ *Пример получения имени пода для сервиса* `evo.siem.collector-manager`

```
kubectl -n evo get pods -l app.kubernetes.io/name=evo.siem.collector-manager
```

Пример вывода команды:

```
NAME                                        READY   STATUS    RESTARTS   AGE
```

```
evo.siem.collector-manager-59dbb7767-hx8nq   1/1     Running   0          43m
```

В этом примере полное имя пода — `evo.siem.collector-manager-59dbb7767-hx8nq`.

В таблице ниже приведены имена и описание функций сервисов, размещенных в пространстве имен кластера. Команды для получения полных имен подов других сервисов содержатся в разделе Поды других сервисов.

Имя сервиса	Функции
`clickhouse`	СУБД ClickHouse.
`dkron`	Запуск задач по расписанию в распределенной среде.
`evo.core.access`	Авторизация в системе, управление пользователями.
`evo.core.bff`	API для интерфейса.
`evo.core.host-app`	Отображение интерфейса системы.
`evo.core.mf-feed`	Конфигурация микрофронтендов.
`evo.siem.active-list-manager`	Управление активными списками.
`evo.siem.active-list-record`	Управление записями в активных списках.
`evo.siem.audit-manager`	Аудит.
`evo.siem.collector-manager`	Управление коллекторами.
`collector-<collector_id>`	Сервис коллектора. Для каждого включенного коллектора создается отдельный сервис. Идентификатор коллектора `<collector_id>` можно найти в поле **ID** карточки коллектора в разделе системы **Ресурсы → Коллекторы**.
`evo.siem.comment-manager`	Управление комментариями.
`evo.siem.cron-service`	Сервис планировщика задач.
`evo.siem.dashboard-manager`	Управление дашбордами.
`evo.siem.documentation`	Отображение пользовательской документации.
`evo.siem.entity-lookup-manager`	Глобальный поиск сущностей.
`evo.siem.entity-relation-manager`	Управление связями между сущностями.
`evo.siem.event-model-manager`	Управление моделями событий.
`evo.siem.event-storage-manager`	Управление хранилищами событий.
`evo.siem.expertise-manager`	Управление элементами экспертизы.
`evo.siem.file-manager`	Управление файлами.
`evo.siem.log-manager`	Управление логами.
`evo.siem.metric-manager`	Управление метриками.
`evo.siem.pipeline-verification`	Валидация конвейеров.
`evo.siem.resource-manager`	Управление ресурсами экспертизы.

Имя сервиса	Функции
`evo.siem.search`	Поиск событий.
`evo.siem.secret-manager`	Управление секретами.
`evo.siem.settings-manager`	Управление настройками.
`evo.siem.smtp-client`	SMTP-клиент электронной почты.
`evo.siem.template-manager`	Управление шаблонами: • точек входа конвейеров; • конечных точек конвейеров; • шин событий; • фильтров; • VRL-трансформаций; • настроек таблиц.

Поды других сервисов

Поды некоторых сервисов находятся вне пространства имен кластера или доступны по селектору, отличному от `app.kubernetes.io/name`. Команды для получения полных имен этих подов представлены в таблице ниже.

Имя сервиса	Функции	Команда для получения полного имени пода
`nats`	Обнаружение сервисов, хранилище "ключ — значение"	`kubectl -n <namespace> get pods -l app.kubernetes.io/name=nats` Здесь: • `<namespace>` — имя пространства имен, в котором установлен кластер.
`metrics-server`	Реализация Kubernetes Metrics API, предоставляющего элементам кластера информацию об использовании вычислительных ресурсов	`kubectl -n kube-system get pods -l app.kubernetes.io/name=metrics-server`
`prometheus`	Сбор и мониторинг метрик кластера	`kubectl -n kube-prometheus-stack get pods -l app.kubernetes.io/name=prometheus`
`promtail`	Централизованный сбор и хранение логов приложений	`kubectl -n kube-prometheus-stack get pods -l app.kubernetes.io/name=promtail`

Настройка оповещений

О настройке оповещений

В состав системы входит система визуализации Grafana, которая используется для мониторинга метрик, собранных с узлов кластера.

Процесс настройки оповещений в Grafana включает следующие этапы:

1. Авторизация в веб-интерфейсе Grafana.
2. Создание правила оповещений с условиями срабатывания.
3. Настройка точки контакта для отправки оповещений (например, Email, Slack, Telegram).
4. Настройка политики оповещений для маршрутизации оповещений на основе их содержания.

Более подробную информацию о настройке можно найти в официальной документации Grafana.

Вход в веб-интерфейс Grafana

Чтобы выполнить вход в веб-интерфейс Grafana:

1. Подключитесь к кластеру с помощью Lens.
2. В левом меню выберите вкладку **Network** → **Services**.
3. Выберите сервис `kube-prometheus-stack-grafana` из списка.
4. В секции **Connection** найдите поле **Ports** и нажмите на указанный в поле порт.
5. Откроется окно браузера с URL формата `localhost:<forwardedPort>`. Добавьте к URL путь `/grafana`, чтобы получить адрес для входа в интерфейс сервиса. Итоговый адрес будет иметь формат: `localhost:<forwardedPort>/grafana`.
6. Чтобы перейти по новому URL, нажмите ENTER. В открывшемся окне авторизации введите логин и пароль для доступа к Grafana.

Учетные данные Grafana

Чтобы получить учетные данные для авторизации в сервисе Grafana:

1. Подключитесь к кластеру с помощью Lens.
2. В левом меню выберите вкладку **Workload** → **Pods**.
3. Выберите под `kube-prometheus-stack-grafana` из списка.
4. В секции **Environment** найдите переменные для доступа к веб-интерфейсу Grafana:
    - `GF_SECURITY_ADMIN_PASSWORD` — пароль пользователя.
    - `GF_SECURITY_ADMIN_USER` — имя пользователя.

Добавление правила оповещения

Правило оповещения в Grafana предназначено для автоматического мониторинга данных и создания оповещений при достижении определенных условий. Оно позволяет задавать параметры для регулярной проверки метрик или логов и инициировать оповещение в случае отклонения от заданных норм.

Выполните следующие шаги для создания правила оповещения:

1. Перейдите в меню **Alerting** → **Alert rules** и выберите опцию для создания нового правила оповещения.
2. В разделе **Set a query and alert condition** выполните следующие действия:
   a. Выберите источник данных (например, **Prometheus**).
   b. Установите временной диапазон для анализа данных (например, `now-10m to now`).
   c. Нажмите **Make this the alert condition** для создания условия срабатывания оповещения.
   d. В разделе **Operations** укажите функции, которые будут применяться к метрикам (например, `Last` и `Strict`).
   e. Установите пороговое значение для срабатывания оповещения (например, `IS ABOVE 0`).
3. Перейдите к разделу **Alert evaluation behavior**:
   a. Установите частоту проверки условия в поле **Evaluate every** (например, `1m`).
   b. Укажите продолжительность, в течение которой условие должно оставаться истинным, чтобы оповещение сработало, в поле **For** (например, `5m`).
4. В разделе **Add details for your alert**:
   a. Укажите название правила оповещения в поле **Rule name**.
   b. Выберите папку и группу для хранения правила в полях **Folder** и **Group**.
   c. Добавьте описание и аннотации к оповещению в полях **Summary**, **Description** и **Runbook URL**. Используйте соответствующие поля для добавления идентификаторов оповещения, панели, или дашборда.
5. В разделе **Notifications**:
   a. Настройте обработку оповещений. Grafana связывает оповещения с точками контактов, используя метки.
   b. Чтобы добавить пользовательские метки, выберите **Custom Labels** и укажите ключ и значение метки. Метки позволяют маршрутизировать оповещения по определенным правилам.
   c. Для добавления новой метки нажмите **Add label** и заполните поля **Choose key** и **Choose value**.
   d. Grafana использует корневой маршрут (**Root route**) по умолчанию для всех оповещений. Если не указаны пользовательские метки, оповещения будут направлены по этому маршруту.
6. Нажмите **Save and exit** для сохранения правила оповещения.

Настройка точек контакта

Используйте точки контакта Grafana для выбора предпочитаемого канала связи для получения оповещений при срабатывании правил оповещений. Вы можете добавлять, редактировать, удалять и тестировать точки контакта.

Точки контакта

На вкладке **Contact Points** вы можете:

- Искать точки контакта по имени и типу.

- Просматривать все существующие точки контакта.
- Добавлять новые точки контакта и редактировать существующие.
- Удалять точки контакта. Обратите внимание, что вы не можете удалить точки контакта, которые используются политикой оповещений. Для этого необходимо либо удалить политику оповещений, либо обновить ее, указав другую точку контакта.

Добавление точки контакта

Выполните следующие шаги для добавления точки контакта:

1. В меню слева перейдите в раздел **Alerting** → **Contact points**
2. В выпадающем меню **Choose Alertmanager** выберите менеджер оповещений. По умолчанию выбран **Grafana**.
3. На вкладке **Contact Points** нажмите **New contact point**.
4. Введите имя точки контакта.
5. Выберите тип точки в поле **Contact point type** и заполните обязательные поля. Например, если вы выбрали **Email**, введите адреса электронной почты через точку с запятой.
6. Некоторые интеграции точек контакта, такие как **Email** или **Webhook**, имеют дополнительные настройки. В разделе **Optional settings** укажите дополнительные параметры для выбранной интеграции точки контакта. Например, можно задать формат заголовка и текст оповещения.
7. В разделе **Notification settings** по желанию выберите **Disable resolved message**, если вы не хотите получать оповещение, когда оповещение будет разрешено.
8. Чтобы добавить еще одну интеграцию точки контакта, нажмите **New contact point type** и повторите шаги 6-8.
9. Сохраните изменения.

Тестирование точки контакта

Выполните следующие шаги для тестирования точки контакта:

1. В меню слева перейдите в раздел **Alerting** → **Contact points**, чтобы просмотреть список существующих точек контакта.
2. На вкладке **Contact Points** найдите точку контакта, которую хотите протестировать, затем нажмите **Edit**. Вы также можете создать новую точку контакта, если это необходимо.
3. Нажмите **Test**, чтобы открыть диалоговое окно тестирования точки контакта.
4. Выберите, отправить ли предварительно определенное тестовое оповещение или выберите **Custom**, чтобы добавить свои собственные аннотации и метки для включения в оповещение.
5. Нажмите **Send test notification**, чтобы инициировать оповещение.

Настройка политики оповещений

Политики оповещений в Grafana управляют маршрутизацией оповещений к точкам контакта. Политики имеют древовидную структуру, где каждая политика может содержать одну или несколько дочерних политик, а также может включать правила для сопоставления меток оповещений. Оповещения оцениваются по умолчанию с помощью корневой политики, после чего

последовательно оцениваются каждой дочерней политикой.

Редактирование корневой политики оповещений

Для настройки корневой политики оповещений в Grafana:

1. В меню слева выберите **Alerting** → **Notification policies**.
2. Убедитесь, что в выпадающем меню **Choose Alertmanager** выбран **Grafana**.
3. В секции **Root policy - default for all alerts** нажмите **Edit**.
4. В поле **Default contact point** выберите точку контакта, куда будут направляться все оповещения, если не установлены другие правила маршрутизации.
5. В секции **Group by** выберите метки, по которым будут группироваться оповещения. Например, можно выбрать метки `grafana_folder` и `alertname`, чтобы группировать оповещения по этим критериям.
6. Разверните секцию **Timing options** и настройте параметры времени:
    a. **Group wait** — время ожидания перед отправкой первого оповещения для новой группы. По умолчанию 30 секунд.
    b. **Group interval** — минимальный интервал времени между отправкой оповещений в группе. По умолчанию 5 минут.
    c. **Repeat interval** — минимальный интервал времени для повторной отправки оповещения, если в группе не было добавлено новых оповещений. По умолчанию 4 часа.
7. Нажмите **Save**, чтобы сохранить изменения.

Добавление дочерней политики

Для создания дочерней политики, которая будет использоваться для определенных меток:

1. В меню слева выберите **Alerting** → **Notification policies**.
2. Убедитесь, что в выпадающем меню **Choose Alertmanager** выбран **Grafana**.
3. В секции **Specific routing** нажмите **New specific policy**.
4. В секции **Matching labels** добавьте одно или несколько правил для соответствия меткам оповещений.
5. В поле **Contact point** выберите точку контакта, куда будут отправляться оповещения, если они соответствуют текущей политике.
6. При необходимости включите опцию **Continue matching subsequent sibling nodes**, чтобы продолжить обработку других дочерних политик после совпадения с текущей.
7. При необходимости включите опцию **Override grouping**, если требуется изменить группировку оповещений.
8. При необходимости включите опцию **Override general timings**, чтобы переопределить временные параметры, настроенные в корневой политике.
9. Нажмите **Save policy**, чтобы сохранить настройки.

Добавление соседней политики

Для создания соседней политики рядом с существующей дочерней политикой:

1. В меню слева выберите **Alerting** → **Notification policies**.
2. Найдите дочернюю политику, для которой требуется создать соседнюю политику.
3. Нажмите **Add new policy** и выберите **New sibling above** или **New sibling below**.
4. Настройте новую политику аналогично дочерней, задав метки, точку контакта и дополнительные параметры.
5. Нажмите **Save policy**, чтобы сохранить настройки.

Поиск политик

Для поиска политик в дереве настроек используйте следующие параметры:

- **Label matchers** — для поиска по меткам.
- **Contact Points** — для поиска по точкам контакта.

Для поиска по точке контакта выберите ее из выпадающего меню **Search by contact point**. Политики, использующие эту точку контакта, будут выделены в интерфейсе.

Для поиска по меткам введите соответствующие значения в поле **Search by matchers**. Можно комбинировать несколько меток, разделяя их запятой.

Настройка времени отключения оповещений

Mute timings используются для указания временных интервалов, в течение которых оповещения должны быть отключены. Эти настройки не наследуются от родительской политики и должны быть настроены на каждом уровне.

Для добавления временных интервалов отключения:

1. В меню слева выберите **Alerting** → **Notification policies**.
2. Нажмите **Add mute timing**.
3. Укажите временные интервалы, в которые оповещения должны быть отключены.
4. Нажмите **Save timing**, чтобы сохранить настройки.

Теперь политики оповещений настроены и будут корректно обрабатывать оповещения в зависимости от заданных параметров.

Настройка оповещений в Telegram

Для настройки оповещений в Telegram в Grafana необходимо выполнить следующие шаги:

1. Создать и настроить Telegram-бота.
2. Подключить Telegram-бота к Grafana.
3. Настроить точку контакта для отправки оповещений через Telegram.
4. Добавить созданную точку контакта в политику уведомлений для маршрутизации оповещений.

Настройка Telegram-бота

Для создания и настройки Telegram-бота:

1. Откройте приложение Telegram на вашем устройстве.
2. Найдите бота с именем **@BotFather**.
3. Отправьте команду **/newbot**.
4. Введите название бота, например, **EVO Notifier**.
5. Введите имя пользователя для бота. Имя должно оканчиваться на "bot" или "_bot", например, **evonotifybot**.
6. Сохраните значение токена, предоставленное в ответе.
7. Создайте групповую беседу или откройте уже существующую в приложении Telegram.
8. Добавьте созданного бота в эту группу, найдя его по имени пользователя.
9. Скопируйте идентификатор чата из URL-адреса в адресной строке браузера. Адрес должен выглядеть примерно так: `https://web.telegram.org/a/#-4266605675`. Идентификатор чата — это последовательность чисел, следующих за символом `#`. Например, `-4266605675`.

Настройка точки контакта в Grafana

После настройки Telegram-бота, выполните следующие шаги для его интеграции с Grafana:

1. Выполните вход в веб-интерфейс Grafana.
2. Перейдите в **Alerting** → **Contact points**.
3. Нажмите **New contact point**, чтобы создать новую точку контакта.
4. Введите имя точки контакта в поле **Name**.
5. В списке **Contact point type** выберите **Telegram**.
6. Укажите настройки Telegram:
   - **BOT API Token** — токен Telegram-бота.
   - **Chat ID** — ID чата в Telegram.
7. В разделе **Optional Telegram settings** (опционально):
   a. Включите опцию **Disable resolved message**, чтобы не получать сообщения о разрешении оповещения.
   b. Включите опцию **Include image**, чтобы добавить изображение в оповещение.
8. Нажмите **Test** для отправки тестового сообщения и проверки работы интеграции.
9. Нажмите **Save contact point**, чтобы сохранить настройки.

Добавление точки контакта в политику уведомлений

После создания точки контакта для оповещений через Telegram необходимо добавить ее в политику уведомлений.

Для добавления точки контакта в политику уведомлений выполните следующие шаги:

1. Перейдите в **Alerting** → **Notification policies**.
2. Убедитесь, что в выпадающем меню **Choose Alertmanager** выбран **Grafana**.
3. В секции **Root policy - default for all alerts** нажмите **Edit**.
4. В поле **Default contact point** выберите созданную ранее точку контакта для оповещений через Telegram.
5. При необходимости в разделе **Group by** выберите метки, по которым будут группироваться оповещения.
6. При необходимости настройте параметры времени в разделе **Timing options**:
    a. **Group wait** — время ожидания перед отправкой первого уведомления для новой группы.
    b. **Group interval** — минимальный интервал времени между отправкой уведомлений в группе.
    c. **Repeat interval** — минимальный интервал времени для повторной отправки уведомления, если в группе не было добавлено новых оповещений.
7. Нажмите **Save**, чтобы сохранить изменения.

После выполнения этих шагов все оповещения, соответствующие установленным критериям, будут отправляться на указанную точку контакта через Telegram.

Настройка оповещений по электронной почте

Для настройки оповещений по электронной почте в Grafana необходимо выполнить следующие шаги:

1. Настроить SMTP-сервер в конфигурационном файле Grafana.
2. Создать и настроить точку контакта для отправки оповещений по электронной почте через веб-интерфейс Grafana.
3. Добавить созданную точку контакта в политику уведомлений для маршрутизации оповещений.

Конфигурация SMTP-сервера

Прежде чем настраивать канал оповещений по электронной почте, необходимо настроить SMTP-сервер в конфигурационном файле Grafana.

Редактирование конфигурационного файла Grafana

Для настройки SMTP сервера в Grafana выполните следующие шаги:

1. Подключитесь к кластеру с помощью Lens.
2. В выпадающем меню свойств кластера выберите пункт **Config** → **Config Maps**.
3. Выберите карту конфигурации **kube-prometheus-stack-grafana** из списка.
4. В открывшейся панели справа найдите файл **grafana.ini** и добавьте следующую секцию для настройки SMTP:

```
[smtp]
enabled = true
host = localhost:25  # Укажите реальный адрес и порт вашего SMTP сервера
```

```
user =  # Имя пользователя (если требуется)
password =  # Пароль (если требуется)
from_address =  # Адрес отправителя
from_name =  # Имя отправителя
skip_verify = true  # Используйте только для тестирования
```

▼ *Пример настроек SMTP*

```
[smtp]
enabled = true
host = smtp.example.com:587
user = yourusername
password = yourpassword
from_address = alert@example.com
from_name = Grafana Alerts
skip_verify = false
```

5. Сохраните изменения.
6. Для применения изменений удалите под Grafana с именем, начинающимся на **kube-prometheus-stack-grafana-** в разделе **Workloads** → **Pods**.

Настройка точки контакта

После настройки SMTP-сервера в конфигурационном файле можно настроить канал оповещений по электронной почте в веб-интерфейсе Grafana.

1. Выполните вход в веб-интерфейс Grafana.
2. Перейдите в раздел **Alerting** → **Contact points**.
3. Нажмите **New contact point**, чтобы создать новую точку контакта.
4. Введите имя точки контакта в поле **Name**.
5. В списке **Contact point type** выберите **Email**.
6. В поле **Addresses** укажите адреса электронной почты получателей (можно указать несколько адресов через разделитель `;`).
7. В разделе **Optional Email settings** (опционально):
   a. Включите опцию **Single email**, чтобы отправить одно письмо всем получателям.
   b. В поле **Message** введите сообщение, которое будет включено в письмо. Можно использовать шаблонные переменные.
   c. В поле **Subject** укажите тему письма. Можно использовать шаблонные переменные.
8. В разделе **Notification settings** (опционально):
   a. Включите опцию **Disable resolved message**, чтобы не получать сообщения о разрешении оповещения.
9. Нажмите **Test** для отправки тестового сообщения и проверки работы интеграции.
10. Нажмите **Save contact point**, чтобы сохранить настройки.

Точка контакта теперь готова к использованию в правилах оповещения.

Добавление точки контакта в политику уведомлений

После настройки точки контакта для оповещений по электронной почте необходимо добавить ее в политику уведомлений, чтобы Grafana могла направлять оповещения на эту точку контакта.

Для добавления точки контакта в политику уведомлений выполните следующие шаги:

1. В меню слева выберите **Alerting** → **Notification policies**.
2. Убедитесь, что в выпадающем меню **Choose Alertmanager** выбран **Grafana**.
3. В секции **Root policy - default for all alerts** нажмите **Edit**.
4. В поле **Default contact point** выберите ранее созданную точку контакта для оповещений по электронной почте.
5. При необходимости в секции **Group by** выберите метки, по которым будут группироваться оповещения. Например, можно выбрать метки `grafana_folder` и `alertname`, чтобы группировать оповещения по этим критериям.
6. При необходимости разверните секцию **Timing options** и настройте параметры времени:
   a. **Group wait** — время ожидания перед отправкой первого уведомления для новой группы. По умолчанию 30 секунд.
   b. **Group interval** — минимальный интервал времени между отправкой уведомлений в группе. По умолчанию 5 минут.
   c. **Repeat interval** — минимальный интервал времени для повторной отправки уведомления, если в группе не было добавлено новых оповещений. По умолчанию 4 часа.
7. Нажмите **Save**, чтобы сохранить изменения.

После выполнения этих шагов все оповещения, которые соответствуют установленным критериям, будут отправляться на указанную точку контакта по электронной почте.

Работа с метриками Prometheus

В дистрибутив системы входит система мониторинга Prometheus, собирающая различные метрики работы кластера. В данном разделе описано, как получить доступ к панели управления Prometheus и просмотреть метрики, собранные системой.

Подключение к Prometheus

Чтобы получить доступ к панели управления Prometheus:

1. Осуществите проброс портов, выполнив на хосте с сетевым доступом к управляющему узлу кластера следующую команду:

   ```
   kubectl port-forward -n kube-prometheus-stack service/kube-prometheus-stack-
   prometheus <local_port>:9090
   ```

   Здесь:

- `<local_port>` — порт на хосте, по которому будет осуществляться доступ к Prometheus.

2. В браузере откройте URL `localhost:<local_port>`, где `<local_port>` — порт с шага 1. Отобразится панель управления Prometheus.

Просмотр списка метрик, получаемых из источников

Источники метрик — это сервисы, из которых Prometheus получает метрики.

Чтобы просмотреть список источников, на панели управления Prometheus перейдите в раздел **Status** → **Targets**. Для каждого источника доступны метки **Labels**, по которым можно фильтровать значения метрик. Например, в качестве метки может выступать пространство имен или под, на котором располагается источник.

Просмотреть список метрик, получаемых Prometheus из всех источников, можно на главной странице панели управления:

- Чтобы просмотреть список всех доступных метрик:
  1. Введите в строку запроса Prometheus следующий запрос:

     ```
     group by(__name__) ({__name__!=""})
     ```

  2. Нажмите на кнопку **Execute** или на клавишу ENTER.
- Чтобы найти метрики по ключевому слову в названии, введите ключевое слово в строку запроса Prometheus. Отобразится список автодополнения, содержащий метрики с указанным ключевым словом в названии.

Просматривать список метрик, получаемых из конкретного источника, необходимо непосредственно на источнике. Для этого выполните следующие действия:

1. Перейдите в раздел Prometheus **Status** → **Targets**.
2. Выберите нужный источник из списка.
3. Получите из столбца **Endpoint** таблицы порт и путь к конечной точке, на которой находится источник метрик.

   Например, для адреса `http://10.233.116.160:8001/metrics` порт — `8001`, а путь — `/metrics`.

4. Получите из столбца **Labels** таблицы пространство имен и под кластера, где находится источник метрик. Пространство имен указано в значении метки `namespace`, а под — в значении метки `pod`.
5. Осуществите проброс портов до источника метрик, выполнив на хосте с сетевым доступом к управляющему узлу кластера следующую команду:

   ```
   kubectl port-forward -n <namespace> pod/<source_pod>
   <source_local_port>:<source_remote_port>
   ```

   Здесь:

- `<namespace>` — пространство имен кластера, в котором находится источник метрик.
- `<source_pod>` — под, на котором находится источник метрик.
- `<source_local_port>` — порт на хосте, по которому будет осуществляться доступ к источнику метрик. Должен отличаться от порта, выбранного для доступа к Prometheus.
- `<source_remote_port>` — порт, на котором находится источник метрик.

6. В браузере откройте URL `localhost:<source_local_port>/<source_path>`.

   Здесь:

   - `<source_local_port>` — порт с предыдущего шага.
   - `<source_path>` — путь к конечной точке источника метрик, полученный на шаге 3.

     Откроется URL, соответствующий источнику метрик.

▼ *Пример получения списка метрик шарда ClickHouse*

1. Перейдите в раздел Prometheus **Status** → **Targets**.
2. Выберите источник `serviceMonitor/kube-prometheus-stack/kube-prometheus-stack-clickhouse/0`.

   Далее рассматривается получение списка метрик для шарда `clickhouse-shard0-0`. Для второго шарда процесс аналогичен.

3. Получите из столбца **Endpoint** таблицы информации об источнике порт и путь конечной точки, отправляющей метрики шарда `clickhouse-shard0-0`. Пусть конечная точка имеет адрес `http://10.0.0.173:8001/metrics`. Тогда:
   - порт `8001`;
   - путь `/metrics`.
4. Получите из столбца **Labels** таблицы пространство имен и под кластера, где находится источник метрик. Пусть пространство имен в значении метки `namespace` — `evo`, а под в значении метки `pod` — `clickhouse-shard0-0`.
5. Осуществите проброс портов до источника метрик, выполнив на хосте с сетевым доступом к управляющему узлу кластера следующую команду:

   ```
   kubectl port-forward -n evo pod/clickhouse-shard0-0 5555:8001
   ```

6. В браузере откройте URL `localhost:5555/metrics`. Отобразится список метрик ClickHouse.

Вывод значений метрик в интерфейсе Prometheus

На главной странице панели управления Prometheus можно выводить значения получаемых системой метрик. Для этого:

1. Введите запрос на языке PromQL в строку запроса Prometheus.
2. Нажмите на кнопку **Execute** или на клавишу ENTER.
3. При необходимости измените представление данных, пользуясь вкладками:

- **Table** — таблица значений метрик в момент времени, заданный в поле ввода **Evaluation time**;
- **Graph** — графическое представление временного ряда значений метрики. Для него можно выбрать интервал времени для отображения, конечный момент времени **End time** и интервал в секундах между точками графика **Res. (s)**.

Метрики, собранные Prometheus, также можно отобразить на дашборде Grafana.

Примеры метрик состояния кластера

Использование системных ресурсов

- Использование процессора:
  - `container_cpu_usage_seconds_total` — время процессора в секундах, потребленное контейнерами кластера.
  - `machine_cpu_cores` — количество логических ядер процессора.

  ▼ *PromQL-запрос потребления ресурсов процессора всеми контейнерами кластера*

  Метрика `container_cpu_usage_seconds_total` имеет тип `counter`, то есть является счетчиком с накоплением. Чтобы вычислить изменение метрики за временной период, воспользуйтесь функцией `rate` языка PromQL:

  ```
  sum(rate(container_cpu_usage_seconds_total{container!=""}[1m])) /
  sum(machine_cpu_cores) * 100
  ```

  - `container_threads` — количество потоков, запущенных внутри контейнера.

  ▼ *PromQL-запрос количества потоков внутри указанного контейнера*

  ```
  container_threads{container="<container_name>"}
  ```

  Здесь:

  - `<container_name>` — имя контейнера.

- Использование оперативной памяти:
  - `container_memory_usage_bytes` — потребление оперативной памяти контейнерами в байтах.
  - `machine_memory_bytes` — размер оперативной памяти в байтах, установленной на машине.

  ▼ *PromQL-запрос потребления оперативной памяти контейнерами на узле*

  ```
  sum(container_memory_usage_bytes{node="<node_name>"}) /
  sum(machine_memory_bytes{node="<node_name>"}) * 100
  ```

  Здесь:

  - `<node_name>` — имя узла кластера.

- Использование дискового пространства:
  - `host_filesystem_used_bytes` — размер используемого дискового пространства в байтах.
  - `host_filesystem_total_bytes` — размер установленного дискового пространства в байтах.

    ▼ *PromQL-запрос процента свободного места в файловой системе*

    ```
    sum(host_filesystem_used_bytes{filesystem="<filesystem>"}) /
    sum(host_filesystem_total_bytes{filesystem="<filesystem>"}) * 100
    ```

    Здесь:

    - `<filesystem>` — имя файловой системы.

- Использование сети:
  - `container_network_transmit_bytes_total` — объем переданных по сети данных в байтах.
  - `container_network_receive_bytes_total` — объем полученных по сети данных в байтах.

    ▼ *PromQL-запрос количества мегабайт, переданных по сети за одну минуту*

    Метрики `container_network_transmit_bytes_total` и `container_network_receive_bytes_total` имеют тип `counter`. Чтобы вычислить изменение метрики за временной период, воспользуйтесь функцией `rate` языка PromQL:

    ```
    sum(rate(container_network_transmit_bytes_total{node="<node_name>"}[1m])) /
    (1024 ^ 2)
    ```

    Здесь:

    - `<node_name>` — имя узла кластера.
    - Байты переведены в мегабайты путем деления на 1024^2.

Метрики ClickHouse

- Общие метрики запросов:
  - `ClickHouseMetrics_Query` — количество выполняющихся запросов.
  - `ClickHouseProfileEvents_FailedQuery` — количество запросов, завершившихся с ошибкой.
  - `ClickHouseProfileEvents_FailedSelectQuery` — количество запросов типа `SELECT`, завершившихся с ошибкой.
  - `ClickHouseProfileEvents_FailedInsertQuery` — количество запросов типа `INSERT`, завершившихся с ошибкой.
  - `ClickHouseMetrics_GlobalThreadActive` — количество потоков, выполняющих задачи, в глобальном пуле потоков ClickHouse.
  - `ClickHouseErrorMetric_ALL` — количество ошибок сервера ClickHouse с последнего перезапуска.

    ▼ *PromQL-запрос суммарного значения метрики по одному шарду ClickHouse*

```
ClickHouseMetrics_Query{pod="<pod_name>"}
```

Здесь:

- `<pod_name>` — имя пода, на котором расположен шард ClickHouse.

Вместо `ClickHouseMetrics_Query` можно подставить любую метрику из списка общих метрик.

▼ *PromQL-запрос значения метрики по всем шардам ClickHouse*

```
sum(ClickHouseMetrics_Query)
```

Вместо `ClickHouseMetrics_Query` можно подставить любую метрику из списка общих метрик.

- Метрики времени выполнения запросов:
  - `ClickHouseProfileEvents_QueryTimeMicroseconds` — время выполнения запросов в микросекундах.
  - `ClickHouseProfileEvents_SelectQueryTimeMicroseconds` — время выполнения запросов типа `SELECT` в микросекундах.
  - `ClickHouseProfileEvents_InsertQueryTimeMicroseconds` — время выполнения запросов типа `INSERT` в микросекундах.
  - `ClickHouseProfileEvents_OtherQueryTimeMicroseconds` — время выполнения запросов всех типов, кроме `SELECT` и `INSERT`, в микросекундах.

  ▼ *PromQL-запрос времени, затраченного на выполнение запросов в течение одной минуты*

  Метрики, описанные в этом разделе, имеют тип `counter`. Чтобы вычислить изменение метрики за временной период, воспользуйтесь функцией `rate` языка PromQL:

  ```
  sum(rate(ClickHouseProfileEvents_QueryTimeMicroseconds[1m])) / (10^6)
  ```

  Здесь:

  - Микросекунды переведены в секунды путем деления на 10^6.

- Метрики подключений к серверу СУБД:
  - `ClickHouseProfileEvents_DistributedConnectionTries` — количество всех попыток подключения к серверу СУБД.
  - `ClickHouseProfileEvents_DistributedConnectionUsable` — количество успешных попыток подключения к серверу СУБД.
  - `ClickHouseProfileEvents_DistributedConnectionFailTry` — количество неудачных попыток подключения к серверу СУБД, завершившихся повторной попыткой соединения.
  - `ClickHouseProfileEvents_DistributedConnectionFailAtAll` — количество неудачных попыток подключения к серверу СУБД, завершившихся после исчерпания повторных попыток.

▼ *PromQL-запрос количества всех попыток подключения к серверу СУБД в течение одной минуты*

Метрики, описанные в этом разделе, имеют тип `counter`. Чтобы вычислить изменение метрики за временной период, воспользуйтесь функцией `rate` языка PromQL:

```
sum(rate(ClickHouseProfileEvents_DistributedConnectionTries[1m]))
```

Управление кластером

В данном разделе описано управление компонентами кластера средствами командной строки и рабочего стола Lens.

По умолчанию для редактирования конфигурации кластера и его компонентов используется текстовый редактор **vim**. Чтобы использовать другой редактор, переопределите переменную окружения `KUBE_EDITOR` в файле `~/.bashrc`.

▼ *Пример изменения текстового редактора по умолчанию на nano*

```
echo "export KUBE_EDITOR=nano" | tee --append ~/.bashrc
source ~/.bashrc
```

Содержание раздела:

- Управление узлами кластера
- Работа с подом collector-manager
- Увеличение размера тома для подов
- Резервное копирование и восстановление PostgreSQL
- Резервное копирование хранилища etcd
- Получение идентификатора сервиса в NATS Service Discovery
- Работа с объектным хранилищем NATS Object Store
  - Работа с паролями в кластере
  - Изменение доменного имени установленного экземпляра системы

Управление узлами кластера

Данный раздел содержит инструкции по работе с узлами кластера. Доступные операции:

- Просмотр состояния узлов кластера
- Добавление узла к кластеру
- Перезапуск узла кластера
- Удаление узла кластера

Просмотр состояния узлов кластера

С помощью командной строки

Чтобы просмотреть состояние узлов кластера, выполните следующую команду:

```
kubectl get nodes -o wide
```

▼ *Пример списка узлов кластера*

```
NAME                                  STATUS   ROLES           AGE    VERSION
INTERNAL-IP   EXTERNAL-IP   OS-IMAGE               KERNEL-VERSION
CONTAINER-RUNTIME
evo-master01.k8s.dev.rvision.local   Ready    control-plane   140d   v1.26.3
10.99.57.12   <none>        RED OS MUROM (7.3.2)   5.15.131-1.el7.3.x86_64
containerd://1.7.0
evo-worker01.k8s.dev.rvision.local   Ready    <none>          140d   v1.26.3
10.99.57.13   <none>        RED OS MUROM (7.3.2)   5.15.131-1.el7.3.x86_64
containerd://1.7.0
evo-worker02.k8s.dev.rvision.local   Ready    <none>          140d   v1.26.3
10.99.57.11   <none>        RED OS MUROM (7.3.2)   5.15.131-1.el7.3.x86_64
containerd://1.7.0
evo-worker03.k8s.dev.rvision.local   Ready    <none>          140d   v1.26.3
10.99.57.10   <none>        RED OS MUROM (7.3.2)   5.15.131-1.el7.3.x86_64
containerd://1.7.0
```

Состояние узлов отображается в столбце `STATUS`. Возможные статусы:

- `Ready`: узел исправен и готов к размещению подов.
- `NotReady`: узел столкнулся с некоторой проблемой, и на нем нельзя разместить под.
- `SchedulingDisabled`: на узле отключено создание новых подов.
- `Unknown`: узел недоступен для работы.

С помощью Lens

Чтобы просмотреть состояние узлов кластера:

1. Подключитесь к кластеру с помощью Lens.
2. Перейдите в раздел **Nodes**. В правой части экрана отобразится таблица узлов кластера. Состояние узлов отобразится в столбце **Conditions**.

Также состояние узлов кластера можно посмотреть в разделе системы **Инструменты → Мониторинг**.

Добавление узла к кластеру

Добавление узла к кластеру производится в два этапа:

1. Подготовительные работы на **узле, добавляемом в кластер**:
   a. Установите утилиты, требуемые для подключения узла к кластеру:

```
sudo dnf install bash-completion vim telnet chrony jq htop tcpdump zip unzip
rsync ebtables ethtool conntrack -y
```

b. Настройте синхронизацию времени с сервером времени:

```
sudo timedatectl set-ntp true
sudo sed -i "/^server /d" /etc/chrony.conf
sudo bash -c 'echo "server <time_server>" >> /etc/chrony.conf'
sudo systemctl enable --now chronyd
sudo systemctl restart chronyd
sudo systemctl status chronyd
chronyc sources
```

Здесь:

- `<time_server>` — адрес корпоративного сервера времени (например, `ntp.rvision.local`).

2. Работы на **управляющем узле**:

a. Внесите в файл `inventory/offline-cluster/inventory.ini` информацию о новом узле:

```
sudo nano inventory/offline-cluster/inventory.ini
```

Внесите информацию об узле в группу `all` и в группу, для которой он предназначен. В качестве образца синтаксиса используйте поля инвентарных данных уже имеющихся в этом файле узлов.

b. Запустите Ansible-плейбуки для добавления узла в кластер:

```
sudo chmod -R 755 inventory/offline-cluster/credentials
sudo chmod 777 inventory/offline-cluster/credentials
ansible-playbook -uuser -b -i inventory/offline-cluster/inventory.ini
sysprep.yml --extra-vars archive_directory=../../.. --extra-vars
sysprep_name=sys-prep-redos
ansible-playbook -uuser -b -i inventory/offline-cluster/inventory.ini
cluster.yml
cd ~
```

c. Отключите offline-репозитории:

```
sudo dnf config-manager --set-disabled deps-offline
sudo dnf config-manager --set-enabled base updates
```

d. Проверьте список узлов в кластере:

```
sudo kubectl get node
```

e. Проверьте, какой набор приложений был развернут с помощью `kubespray`:

```
sudo kubectl get po -A
```

Перезапуск узла кластера

При выполнении работ по обслуживанию узла кластера (например, обновлению ОС или добавлению вычислительных ресурсов на хост) может понадобиться перезапуск узла. Чтобы сделать это, выполните на узле следующие команды:

1. Переведите узел в `drain`-режим:

```
kubectl drain <node_name> --ignore-daemonsets
```

Здесь:

- `<node_name>` — имя узла (например, `evo-worker05.k8s.dev.rvision.local`).

Операция `drain` позволяет перераспределить все поды с узла. В процессе выполнения `drain` узел помечается как `unschedulable`. Это предотвращает появление на нем новых подов. Затем `drain` начинает перемещать поды с узла и завершает работу контейнеров, которые на данный момент запущены на узле, отправляя сигнал `TERM` контейнерам в поде.

2. Остановите сервисы `kubelet` и `containerd`:

```
sudo systemctl stop kubelet
sudo systemctl stop containerd
```

3. Произведите обслуживание узла.
4. Повторно запустите сервисы `kubelet` и `containerd`:

```
sudo systemctl start containerd
sudo systemctl start kubelet
```

5. Сделайте узел вновь назначаемым, чтобы на нем появлялись поды:

```
kubectl uncordon <node_name>
kubectl get nodes -o wide
```

Здесь:

- `<node_name>` — имя узла (например, `evo-worker05.k8s.dev.rvision.local`).

Удаление узла кластера

Для удаления узла выполните следующие шаги:

1. Откройте файл инвентаризации:

   ```
   sudo nano inventory/offline-cluster/inventory.ini
   ```

   Удалите узел из файла инвентаризации и сохраните его.

2. Переведите узел в `drain`-режим, чтобы освободить ресурсы на нем:

   ```
   kubectl drain --ignore-daemonsets worker02.example.local
   ```

3. После того как ресурсы будут перераспределены, запустите плейбук на удаление узлов:

   Так как плейбуку требуется доступ к файлу `/etc/kubernetes/admin.conf` и привилегированный доступ, эти шаги необходимо производить от имени пользователя root.

   ```
   ansible-playbook --ask-pass -uroot -i inventory/offline-cluster/inventory.ini
   remove-node.yml --extra-vars "node=worker02.example.local"
   ```

   Плейбук запросит подтверждение операции.

4. После завершения работы плейбука запросите список узлов и проверьте, что узел удален:

   ```
   kubectl get nodes
   ```

Работа с подом collector-manager

В данном разделе описана работа с подом `collector-manager`, который управляет всеми коллекторами в системе. Анализ логов и событий этого пода помогает в диагностике и устранении проблем, связанных с работой коллекторов.

Получение полного имени пода collector-manager

Каждому поду присваивается случайный идентификатор, добавляемый к имени сервиса, для которого создается под.

Чтобы получить полное имя пода `collector-manager`, выполните следующую команду:

```
kubectl -n <namespace> get pods -l app.kubernetes.io/name=evo.siem.collector-manager
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.

▼ *Пример вывода имени пода* `collector-manager`

```
NAME                                          READY   STATUS    RESTARTS   AGE
evo.siem.collector-manager-59dbb7767-hx8nq    1/1     Running   0          43m
```

Опция `-l` команды `kubectl get pods` позволяет отфильтровать список подов кластера по значению селектора `app.kubernetes.io/name`.

В примере выше полное имя пода `collector-manager` — `evo.siem.collector-manager-59dbb7767-hx8nq`.

Просмотр логов пода collector-manager

Логи пода очищаются при его остановке.

Для просмотра логов пода `collector-manager` используйте руководство **Просмотр состояния подов кластера**. В качестве имени пода (`<pod_name>`) укажите полное имя пода `collector-manager`, полученное ранее.

Изменение уровня логирования collector-manager

Если текущие логи пода `collector-manager` содержат избыточную или недостаточную информацию, уровень логирования можно просмотреть и изменить с помощью командной строки. Доступны следующие уровни логирования:

- `error` — записываются критические ошибки, которые могут препятствовать работе коллекторов.
- `warn` — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
- `info` — записывается общая информация о работе и состоянии коллекторов.
- `debug` — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем.
- `trace` — самый подробный уровень логирования, предоставляющий информацию о каждом шаге работы коллекторов.

Просмотр текущего уровня логирования

Чтобы просмотреть текущий уровень логирования:

1. Получите полное имя пода `collector-manager`.
2. Выполните следующую команду:

   ```
   kubectl exec -n <namespace> <pod_name> -- node -e
   "console.log(process.env.SIEM_COLLECTOR_MANAGER_LOG_LEVEL)"
   ```

   Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<pod_name>` — полное имя пода `collector-manager`, полученное на шаге 1.

Изменение уровня логирования

Чтобы изменить уровень логирования:

1. Получите полное имя пода `collector-manager`.
2. Отредактируйте карту конфигурации `collector-manager`:
   a. Откройте карту конфигурации, выполнив следующую команду:

      ```
      kubectl -n <namespace> edit configmap evo.siem-collector-manager-env
      ```

      Здесь:

      - `<namespace>` — имя пространства имен, в котором установлен кластер.

   b. Укажите требуемый уровень логирования (`error`, `warn`, `info`, `debug` или `trace`) как значение ключа `SIEM_COLLECTOR_MANAGER_LOG_LEVEL`.
   c. Сохраните файл.
   d. Перезагрузите деплоймент `evo.siem.collector-manager`, чтобы применить изменения:

      ```
      kubectl -n <namespace> rollout restart deployment evo.siem.collector-manager
      ```

      Здесь:

      - `<namespace>` — имя пространства имен, в котором установлен кластер.

Увеличение размера тома для подов

В данном разделе описано, как увеличить размер тома пода через изменение PVC (Persistent Volume Claims).

С помощью командной строки

1. Чтобы просмотреть список всех PVC в кластере, выполните следующую команду:

   ```
   kubectl get pvc -n <namespace>
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.

   ▼ *Пример списка PVC в кластере*

   ```
   NAME                                                    STATUS    VOLUME
   CAPACITY   ACCESS MODES   STORAGECLASS      AGE
   ```

```
collector-0213f945-3587-421e-9085-83989a881cba              Bound     pvc-
18c76f8e-6573-490f-889b-09048028abbc   64Mi       RWO            piraeus-
storage   141d
collector-04d59675-aaee-4d4b-ae88-3be312bcf92e              Bound     pvc-
203aa4dc-ad24-4e46-b220-bc2a1652f602   64Mi       RWO            piraeus-
storage   148d
collector-063a7205-f8b6-4c56-bb19-991a73326433              Bound     pvc-
2bfb626d-81c4-4553-93f9-aa85be36f214   64Mi       RWO            piraeus-
storage   141d
collector-0713a9d1-2293-46c1-9df7-9f84ebb65fa9              Bound     pvc-
3f36e415-4bd1-4a56-a710-c07162e13841   64Mi       RWO            piraeus-
storage   142d
collector-08712a52-f8c2-4dd2-b4a3-ecc5e841601e              Bound     pvc-
2f979aab-aaa6-4ef2-9879-93e4f47916aa   64Mi       RWO            piraeus-
storage   143d
collector-0917d12f-be94-4205-8b27-87c15d975c06              Bound     pvc-
a1581417-bb0e-4cd4-8a7a-add95a33fd7c   64Mi       RWO            piraeus-
storage   143d
collector-094eea63-a56a-4813-9d8e-2eea21909e38              Bound     pvc-
2793d933-dcf2-43d6-a5d9-4e08ae301ece   64Mi       RWO            piraeus-
storage   147d
collector-0a6a0b2f-65fb-4f4a-b774-87405999699c              Bound     pvc-
ea316cf7-cb92-476d-9f0d-af31ec2e6798   64Mi       RWO            piraeus-
storage   134d
collector-0c9f9fab-df58-4322-aced-fa8c3521e7fb              Bound     pvc-
0e7d779d-b5f0-479d-a151-63cc19d834ad   64Mi       RWO            piraeus-
storage   140d
collector-0cfe4aea-40e6-412b-b82f-22f3a28ab6bc              Bound     pvc-
7a669403-e9bb-4e30-a2a7-5ccf89f00135   64Mi       RWO            piraeus-
storage   148d
collector-10fd33b9-b05f-4047-b56f-2e19f6798e77              Bound     pvc-
64484986-ef17-4d84-bee1-67a0241b8450   64Mi       RWO            piraeus-
storage   146d
collector-12fc0ffb-fe27-4741-ac09-e8deba114aa9              Bound     pvc-
5901accc-62ce-4e33-8427-051c7b184dab   64Mi       RWO            piraeus-
storage   146d
collector-14277d5d-b2ac-4c7f-a14e-261aa5345751              Bound     pvc-
1a5d6627-379e-4fae-89e2-2e160481cdca   64Mi       RWO            piraeus-
storage   134d
collector-14277d5d-b2ac-4c7f-a14e-261aa5345751-correlator0   Pending
piraeus-storage   134d
collector-14277d5d-b2ac-4c7f-a14e-261aa5345751-correlator1   Pending
piraeus-storage   134d
```

Для получения подробной информации о PVC, такую как таблица событий PVC (`Events`), выполните следующую команду:

```
kubectl describe pvc <pvc_name>
```

Здесь:

- `<pvc_name>` — имя PVC.

▼ *Пример вывода команды kubectl describe pvc <pvc_name>*

```
Name:          test-pvc
Namespace:     evo
StorageClass:  piraeus-storage
Status:        Bound
Volume:        pvc-6252c148-eb98-47ce-9e1a-e13a3ae3a45d
Labels:        app.kubernetes.io/instance=test-pvc
               app.kubernetes.io/managed-by=Helm
               app.kubernetes.io/name=test-pvc
               app.kubernetes.io/version=1.16.0
               helm.sh/chart=test-pvc-0.1.0
Annotations:   meta.helm.sh/release-name: test-pvc
               meta.helm.sh/release-namespace: evo
               pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-
provisioner: linstor.csi.linbit.com
               volume.kubernetes.io/selected-node: evo-
worker02.k8s.dev.rvision.local
               volume.kubernetes.io/storage-provisioner:
linstor.csi.linbit.com
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      64Mi
Access Modes:  RWO
VolumeMode:    Filesystem
Used By:       test-pvc-6765945674-cgbn2
Events:
  Type    Reason                 Age                 From
Message
  ----    ------                 ----                ----
-------
  Normal  VolumeConditionNormal  16m (x12 over 71m)  csi-
pv-monitor-controller-linstor.csi.linbit.com  The Volume
returns to the healthy state
```

2. Чтобы открыть файл конфигурации PVC, выполните следующую команду:

```
kubectl edit pvc <pvc_name>
```

Здесь:

- `<pvc_name>` — имя PVC.

3. В открывшемся файле задайте новый размер PVC в ключе `spec → resources → requests → storage`:

```
# ...
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      # отредактируйте значение этого ключа
      storage: 64Mi
# ...
```

4. Закройте файл. Kubernetes перераспределит дисковое пространство в соответствии с обновленным PVC, и размер тома изменится через некоторое время.

С помощью Lens

1. Подключитесь к кластеру с помощью Lens.
2. В выпадающем меню свойств кластера выберите пункт **Storage** → **Persistent Volume Claims**.
3. Выберите пространство имен кластера в выпадающем списке **Select Namespace**. Отобразится таблица PVC.
4. Выберите нужный PVC в списке. В правой части экрана отобразится панель свойств PVC.
5. Нажмите на кнопку **Edit** в правом верхнем углу экрана. В нижней части экрана отобразится файл конфигурации PVC.
6. Отредактируйте значение ключа `spec → resources → requests → storage` аналогично п. 3 инструкции выше.
7. Закройте файл. Kubernetes перераспределит дисковое пространство в соответствии с обновленным PVC, и размер тома изменится через некоторое время.

Резервное копирование и восстановление PostgreSQL

Резервная копия базы данных системы может понадобиться для ее восстановления после сбоя или переноса на другое оборудование. Система использует СУБД PostgreSQL для хранения конфигураций сервисов.

В данном разделе рассматривается процесс резервного копирования, когда инсталляция PostgreSQL находится внутри кластера.

Резервное копирование

Чтобы создать резервную копию таблиц PostgreSQL, выполните следующие действия:

1. Подключитесь к управляющему узлу кластера.
2. В любой директории создайте скрипт backup_pg.sh со следующим содержимым:

▼ *backup_pg.sh*

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

```
# Создание директории для сохранения резервной копии.
back_dir=<backup-dir>
mkdir -p $back_dir

# Создание резервной копии всех баз данных.
kubectl exec -n <namespace> postgresql-0 -- /bin/sh -c 'pg_dumpall --port 5432
--username postgres -w --verbose --file "/tmp/all_databases.sql"'

# Проверка, что файл был создан.
kubectl exec -n <namespace> postgresql-0 -- /bin/sh -c 'test -f
/tmp/all_databases.sql'
if [ $? -ne 0 ]; then
    echo "Ошибка: файл /tmp/all_databases.sql не был создан"
    exit 1
fi

# Копирование резервной копии из контейнера на хост.
kubectl cp -n <namespace> postgresql-0:/tmp/all_databases.sql
$back_dir/all_databases.sql

# Архивирование резервной копии.
cd $back_dir
tar -czpf postgres_backup_all_databases.tar.gz all_databases.sql

# Удаление временного файла.
kubectl exec -n <namespace> postgresql-0 -- /bin/sh -c 'rm
/tmp/all_databases.sql'

echo "Резервная копия всех баз данных создана и сохранена в
$back_dir/postgres_backup_all_databases.tar.gz"
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<backup-dir>` — путь к директории, в которой будет создана резервная копия.

3. Разрешите выполнение созданного скрипта:

```
chmod +x backup_pg.sh
```

4. Запустите скрипт:

```
./backup_pg.sh
```

В результате выполнения скрипта по указанному пути `<backup-dir>` будет создан архив postgres_backup_all_databases.tar.gz с резервной копией всех баз данных PostgreSQL..

Восстановление из резервных копий

Чтобы восстановить таблицы PostgreSQL из резервной копии, выполните следующие действия:

1. Подключитесь к управляющему узлу кластера.
2. В любую директорию загрузите архив postgres_backup_all_databases.tar.gz, полученный в результате резервного копирования.
3. В этой же директории создайте скрипт restore_pg.sh со следующим содержимым:

▼ *restore_pg.sh*

```
#!/bin/bash

backup_file="$(dirname "$0")/postgres_backup_all_databases.tar.gz"
extract_dir="/tmp/pg_conf/restore"
mkdir -p $extract_dir

# Список баз данных, которые НЕ нужно удалять (исключения для удаления).
# Не следует редактировать, изменения могут привести к ошибкам в работе системы.
excluded_databases=("postgres" "template0" "template1" "evo_core_access"
"evo_core_mf_feed")

# Список баз данных, которые НЕ нужно восстанавливать (исключения для
восстановления).
# Не следует редактировать, изменения могут привести к ошибкам в работе системы.
excluded_restore_databases=("evo_core_access" "evo_core_mf_feed")

# Проверка наличия файла резервной копии.
if [ ! -f "$backup_file" ]; then
    echo "Ошибка: файл резервной копии $backup_file не найден"
    exit 1
fi

echo "Распаковка архива $backup_file"
tar -xzf "$backup_file" -C "$extract_dir"

# Проверка, что файл all_databases.sql существует.
if [ ! -f "$extract_dir/all_databases.sql" ]; then
    echo "Ошибка: файл all_databases.sql не найден в архиве"
    exit 1
fi

echo "Копирование файла резервной копии в под"
kubectl cp -n <namespace> "$extract_dir/all_databases.sql" postgresql-
0:/tmp/all_databases.sql

echo "Завершение всех активных сессий для баз данных, которые нужно удалить"
kubectl exec -n <namespace> postgresql-0 -- /bin/sh -c "
  psql --port 5432 --username postgres -w -tAc \"
    SELECT pg_terminate_backend(pg_stat_activity.pid)
    FROM pg_stat_activity
```

```
    WHERE pg_stat_activity.datname NOT IN ('$(IFS="','"; echo
"${excluded_databases[*]}")')
    AND pid <> pg_backend_pid();
  \"
"

echo "Удаление всех баз данных, кроме исключений"
kubectl exec -n <namespace> postgresql-0 -- /bin/sh -c "
  psql --port 5432 --username postgres -w -tAc \"
    SELECT 'DROP DATABASE IF EXISTS ' || quote_ident(datname) || ';'
    FROM pg_database
    WHERE datistemplate = false
    AND datname NOT IN ('$(IFS="','"; echo "${excluded_databases[*]}")');
  \" | psql --port 5432 --username postgres -w
"

echo "Удаление всех пользовательских ролей, кроме исключений"
kubectl exec -n <namespace> postgresql-0 -- /bin/sh -c "
  psql --port 5432 --username postgres -w -tAc \"
    SELECT 'DROP ROLE IF EXISTS ' || quote_ident(rolname) || ';'
    FROM pg_roles
    WHERE rolname NOT IN (
      'postgres', 'pg_read_all_data', 'pg_write_all_data',
      'pg_monitor', 'pg_read_all_settings', 'pg_read_all_stats',
      'pg_stat_scan_tables', 'pg_read_server_files', 'pg_write_server_files',
      'pg_execute_server_program', 'pg_signal_backend'
    );
  \" | psql --port 5432 --username postgres -w
"

echo "Фильтрация файла резервной копии для исключения определенных баз данных"
kubectl exec -n <namespace> postgresql-0 -- /bin/sh -c "
  grep -Ev '^(CREATE DATABASE|ALTER DATABASE) ($(IFS="|"; echo
"${excluded_restore_databases[*]}"))' /tmp/all_databases.sql >
/tmp/filtered_databases.sql
"

echo "Создание отсутствующих баз данных перед восстановлением"
kubectl exec -n <namespace> postgresql-0 -- /bin/sh -c "
  psql --port 5432 --username postgres -w -tAc \"
    SELECT 'CREATE DATABASE ' || quote_ident(datname) || ';'
    FROM pg_database
    WHERE datname NOT IN ('$(IFS="','"; echo "${excluded_databases[*]}")')
    AND datname NOT IN (SELECT datname FROM pg_database);
  \" | psql --port 5432 --username postgres -w
"

echo "Восстановление данных из отфильтрованного файла резервной копии"
kubectl exec -n <namespace> postgresql-0 -- /bin/sh -c 'psql --port 5432
--username postgres -w -f /tmp/filtered_databases.sql'
```

```
echo "Очистка временных файлов"
rm -rf "$extract_dir"
kubectl exec -n <namespace> postgresql-0 -- /bin/sh -c 'rm /tmp/all_databases.sql
/tmp/filtered_databases.sql'

echo "Восстановление завершено"
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.

4. Разрешите выполнение созданного скрипта:

```
chmod +x restore_pg.sh
```

5. Запустите скрипт:

```
./restore_pg.sh
```

В результате выполнения скрипта конфигурация сервисов системы будет восстановлена.

Резервное копирование хранилища etcd

В этом разделе описано, как осуществлять резервное копирование и восстановление из копии хранилища конфигураций etcd с помощью утилит `etcdctl` и `etcdutl`.

В разделе рассматривается работа с etcd при следующих условиях:

- кластер развернут с помощью Kubespray;
- etcd установлено как системный сервис и управляется с помощью утилиты `systemctl`.

В системе должны быть установлены утилиты `etcdctl` и `etcdutl` версии не ниже 3.5.17. Инструкция по установке представлена в репозитории разработчика etcd.

Резервное копирование etcd

Чтобы создать резервную копию etcd, выполните следующую команду:

```
ETCDCTL_API=3 etcdctl snapshot save snapshot.db --cert=/etc/ssl/etcd/ssl/ca.pem
--key=/etc/ssl/etcd/ssl/ca-key.pem
```

На конкретных инсталляциях системы пути к файлам TLS-сертификата (`--cert`) и ключа (`--key`) могут отличаться. Просмотреть актуальные пути можно в файле `/etc/etcd.env` как значения следующих переменных окружения:

- `ETCD_CERT_FILE` — путь к файлу сертификата;

- `ETCD_KEY_FILE` — путь к файлу ключа.

Чтобы получить расширенный лог сообщений о ходе создания резервной копии, добавьте к команде выше опцию `--debug=true`.

Чтобы удостовериться, что резервная копия создана успешно, выполните следующую команду:

```
etcdutl snapshot status snapshot.db -w table
```

▼ *Пример вывода команды* `etcdutl snapshot status snapshot.db -w table`

```
+---------+----------+------------+------------+
|  HASH   | REVISION | TOTAL KEYS | TOTAL SIZE |
+---------+----------+------------+------------+
| 7ef846e |   485261 |      11642 |      94 MB |
+---------+----------+------------+------------+
```

Номер ревизии (`REVISION`) увеличивается на 1 при каждой операции записи в etcd.

Восстановление etcd из резервной копии

Для восстановления etcd из резервной копии в кластере Kubernetes следует использовать вариант восстановления "revision bumps": контроллеры Kubernetes используют так называемые информеры (informers), которые работают как кэш. Восстановление etcd из резервной копии без использования revision bumps может привести к некорректному восстановлению кэшей и, как следствие, неопределенному поведению контроллеров.

Чтобы выполнить восстановление etcd из резервной копии:

1. Выполните следующую команду:

   ```
   etcdutl snapshot restore snapshot.db --bump-revision <n_bumps> --mark-compacted
   --data-dir <new_dir>
   ```

   Здесь:

   - `<n_bumps>` — 64-битное целое число, на которое следует увеличить `REVISION`. Его можно оценить, умножив среднее число операций записи в etcd в секунду на число секунд с момента создания резервной копии. Можно установить значение, большее оценки, меньшее — не рекомендуется.

     Например, если в etcd производится 100 операций записи в секунду, а резервная копия создана 24 часа назад, минимально допустимое значение `n_bumps` — 24 * 60 * 100 = 144000.

   - `<new_dir>` — новый каталог данных etcd, например, `/var/lib/etcd_restored/`.

Опция `--mark-compacted` отмечает предыдущие ревизии etcd как сжатые. Это позволяет удостовериться, что все процессы отслеживания etcd

остановлены, и кэши информеров сделаны недействительными.

2. Остановите сервис etcd, выполнив следующую команду:

```
systemctl stop etcd.service
```

3. В файле `/etc/etcd.env` измените значение переменной `ETCD_DATA_DIR` на путь, указанный в качестве значения `<new_dir>` на шаге 1.
4. Запустите сервис etcd, выполнив следующую команду:

```
systemctl start etcd.service
```

5. При необходимости удалите каталог, в котором хранились данные etcd до восстановления, с помощью следующей команды:

```
rm -rf <old_dir>
```

Здесь:

- `<old_dir>` — каталог, в котором хранились данные etcd до восстановления, например, `/var/lib/etcd/`.

Получение идентификатора сервиса в NATS Service Discovery

Автоматическое обнаружение сервисов необходимо для поддержания корректной работы системы при различных конфигурациях кластера, в которых она может быть установлена. Данное руководство описывает получение идентификатора сервиса в компоненте обнаружения сервисов NATS Service Discovery на примере ClickHouse.

Чтобы получить идентификатор ClickHouse в NATS Service Discovery, выполните следующие действия:

1. Подключитесь к терминалу управляющего узла кластера.
2. Подключитесь к терминалу пода `nats-main-box`, выполнив следующую команду:

```
kubectl exec -itn <namespace> deployment/nats-main-box -c nats-box -- sh
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.

3. Получите идентификатор БД ClickHouse из регистра Service Discovery, выполнив следующую команду:

```
nats kv ls r-space-service-discovery-registry
```

Пример 11. Пример вывода идентификаторов сервисов

```
space-main.nats-space
space-main.evo-clickhouse1
```

Искомый идентификатор содержит подстроку `clickhouse`; в примере — `space-main.evo-clickhouse1`.

Работа с объектным хранилищем NATS Object Store

NATS Object Store — это объектное хранилище, используемое для хранения файлов конфигурации различных объектов системы, например, коллекторов, таблиц обогащения и секретов. Эти конфигурации можно экспортировать из NATS Object Store.

Подключение к NATS Object Store

Чтобы получить доступ к терминалу NATS Object Store, выполните следующие действия:

1. Подключитесь к терминалу управляющего узла кластера.
2. Подключитесь к терминалу пода `nats-main-box`, выполнив следующую команду:

   ```
   kubectl exec -itn <namespace> deployment/nats-main-box -c nats-box -- sh
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.

Экспорт конфигураций из NATS Object Store

Чтобы экспортировать конфигурацию объекта из NATS Object Store:

1. Найдите идентификатор (ID) объекта в графическом интерфейсе системы.
2. Откройте терминал NATS Object Store.
3. Получите список объектов в бакете `r-space-global-objects`, выполнив команду:

   ```
   nats object ls r-space-global-objects
   ```

4. Выгрузите содержимое ключей, содержащих ID с шага 1. Для этого выполните следующую команду для каждого из ключей:

   ```
   nats object get r-space-global-objects <key> --output <out_file>
   ```

   Здесь:

   - `<key>` — ключ из бакета `r-space-global-objects`.
   - `<out_file>` — путь к файлу внутри контейнера `nats-box`, в который будет записано

содержимое ключа.

▼ *Пример выгрузки конфигурации коллектора*

Пусть имеется коллектор с ID `4dedef79-c3e5-49ee-93f5-e1d31e408e96`. В бакете есть четыре ключа, содержащие этот ID:

- `evo.collector/4dedef79-c3e5-49ee-93f5-e1d31e408e96/global-functions/`;
- `evo.collector/4dedef79-c3e5-49ee-93f5-e1d31e408e96/event-source/ad67fcb9-f234-476b-b71a-7bc2609eb9b7/event-sources/`;
- `evo.collector/4dedef79-c3e5-49ee-93f5-e1d31e408e96/event-source/ad67fcb9-f234-476b-b71a-7bc2609eb9b7/event-sources-distributed-collector/`;
- `evo.collector/4dedef79-c3e5-49ee-93f5-e1d31e408e96/event-source/ad67fcb9-f234-476b-b71a-7bc2609eb9b7/event-sources-distributed-correlator/`.

Для получения полной конфигурации коллектора необходимо выгрузить содержимое всех четырех ключей.

При необходимости вы также можете выгрузить только определенные элементы конфигурации. Например, чтобы получить конфигурацию глобальных функций, выгрузите значение ключа `evo.collector/4dedef79-c3e5-49ee-93f5-e1d31e408e96/global-functions/`.

5. Содержимое ключей на предыдущем шаге будет выгружено в файловую систему контейнера `nats-box`. Чтобы выгрузить их в файловую систему управляющего узла кластера, выполните следующие действия:

   a. В терминале управляющего узла кластера получите полное имя пода сервиса `nats-main-box`, выполнив следующую команду:

   ```
   kubectl -n <namespace> get pods -l app.kubernetes.io/instance=nats-main | grep
   nats-main-box
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.

   Пример 12. Пример вывода имени пода `nats-main-box`

   ```
   nats-main-box-658b9fc759-h4wzm   1/1     Running   0          10d
   ```

   В примере выше полное имя пода `nats-main-box` — `nats-main-box-658b9fc759-h4wzm`.

   b. Скопируйте файлы конфигурации из пода `nats-main-box` на управляющий узел кластера, выполнив следующую команду:

   ```
   kubectl -n <namespace> cp <nats_main_box_full_name>:<out_file> <target_out_file>
   ```

   Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<nats_main_box_full_name>` — полное имя пода `nats-main-box`.
- `<out_file>` — имя файла внутри контейнера `nats-box`, в который было записано содержимое ключа конфигурации на шаге 4.
- `<target_out_file>` — имя файла на управляющем узле кластера, в который нужно скопировать содержимое ключа.

Работа с паролями в кластере

При изменении паролей от систем хранения и баз данных нужно поменять соответствующие секреты в Kubernetes (Kubernetes Secrets) — объекты, которые хранят конфиденциальную информацию, такую как пароли и токены. Может также потребоваться перезапуск соответствующих приложений, чтобы они смогли повторно прочитать актуальные данные.

В случае с ClickHouse регистрационные данные обновляются через NATS Key/Value — сервис, который хранит конфигурационные данные и позволяет централизованно управлять паролями и настройками.

Смена пароля пользователя ClickHouse

Чтобы обновить пароль пользователя ClickHouse:

1. Измените пароль ClickHouse, выполнив следующий запрос в консоли ClickHouse:

   Пример 13. SQL-запрос

   ```
   ALTER USER <user> IDENTIFIED BY '<new_password>';
   ```

   Здесь:

   - `<user>` — имя пользователя.
   - `<new_password>` — новый пароль.
2. Если требуется изменить пароль пользователя writer, обновите пароль в блоке `<remote_servers>` в конфигурации кластера.
3. Подключитесь к терминалу управляющего узла кластера.
4. Подключитесь к терминалу пода `nats-main-box`, выполнив следующую команду:

   ```
   kubectl exec -itn <namespace> deployment/nats-main-box -c nats-box -- sh
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.
5. Получите пути к ключам, в которых хранятся пароли ClickHouse, выполнив следующую команду:

   ```
   nats kv ls siem
   ```

▼ *Пример путей к ключам, в которых хранятся пароли ClickHouse*

```
space-main/evo-clickhouse1/owner
space-main/evo-clickhouse1/reader
space-main/evo-clickhouse1/writer
```

6. Измените значение пароля по ключу, где он хранится, выполнив следующую команду:

   ```
   nats kv put siem <path_to_key> <new_password>
   ```

   Здесь:

   - `<path_to_key>` — путь к ключу, где хранится пароль.
   - `<new_password>` — новый пароль.

7. Проверьте, что пароль изменился, выполнив следующую команду:

   ```
   nats kv get siem <path_to_key> --raw
   ```

   Здесь:

   - `<path_to_key>` — путь к ключу, где хранится пароль.

Смена пароля PostgreSQL

Чтобы обновить пароль PostgreSQL:

1. Подключитесь к кластеру с помощью Lens.
2. Перейдите в раздел **Workloads** → **Pods**.
3. Выберите под PostgreSQL и нажмите на кнопку терминала.

Также вы можете подключиться к контейнеру PostgreSQL с помощью командной строки:

```
kubectl exec -it -n <namespace> <postgresql_container> -- sh
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<postgresql_container>` — имя контейнера PostgreSQL.

4. Подключитесь к базе данных.

   ```
   psql -U postgres -d auth
   ```

5. Выполните запрос для обновления пароля.

   Пример 14. SQL-запрос

   ```
   ALTER USER <postgres_user> WITH PASSWORD '<new_password>';
   ```

   Здесь:

   - `<postgres_user>` — имя пользователя.
   - `<new_password>` — новый пароль.

   После смены пароля сервисы, использующие PostgreSQL, перестанут работать до обновления секретов в Kubernetes.

6. Поменяйте секреты Kubernetes, содержащие пароль, выполнив следующие действия для каждого секрета:

   a. Откройте секрет для редактирования:

      ```
      kubectl -n <namespace> edit secret <secret_name>
      ```

      Здесь:

      - `<namespace>` — имя пространства имен, в котором установлен кластер.
      - `<secret_name>` — имя секрета: `evo.core.global` и `evo.siem.global`.

   b. Закодируйте пароль в Base64, например, командой:

      ```
      echo -n <new_password> | base64
      ```

      Здесь:

      - `<new_password>` — новый пароль.

   c. Замените в секрете значение ключа `POSTGRES_PASSWORD` на пароль, закодированный на предыдущем шаге.

   d. Проверьте, что значение ключа было обновлено, командой:

      ```
      kubectl -n <namespace> get secret <secret_name>
      ```

      Здесь:

      - `<namespace>` — имя пространства имен, в котором установлен кластер.
      - `<secret_name>` — имя секрета: `evo.core.global` и `evo.siem.global`.

7. Перейдите в **Workloads** → **Deployments** и перезапустите деплойменты кластера:

   - `evo.core.access`;

- evo.core.mf-feed;
- evo.siem.active-list-manager;
- evo.siem.dashboard-manager;
- evo.siem.driver-manager;
- evo.siem.entity-relation-manager;
- evo.siem.event-model-manager;
- evo.siem.event-storage-manager;
- evo.siem.expertise-manager;
- evo.siem.integration-manager;
- evo.siem.query-log-manager;
- evo.siem.secret-manager;
- evo.siem.settings-manager;
- evo.siem.template-manager.

Либо выполните в командной строке команду для каждого деплоймента:

```
kubectl -n <namespace> rollout restart deployment <deployment_name>
```

Здесь:

- <namespace> — имя пространства имен, в котором установлен кластер.
- <deployment_name> — имя деплоймента.

8. Проверьте состояние подов:

```
kubectl -n <namespace> get pods
```

Здесь:

- <namespace> — имя пространства имен, в котором установлен кластер.

Изменение доменного имени установленного экземпляра системы

Чтобы изменить доменное имя уже установленного экземпляра системы без потери имеющихся в системе данных, выполните следующие действия с кластером Kubernetes:

1. Измените имена хостов для объектов Ingress, входящих в кластер:
   a. Получите список объектов Ingress в кластере:

   ```
   kubectl -n <namespace> get ingress
   ```

   Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.

b. Отредактируйте имя хоста для каждого объекта Ingress в списке, используя команду:

```
kubectl -n <namespace> edit ingress <ingress_name>
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<ingress_name>` — имя объекта Ingress, для которого вы редактируете конфигурацию.

В открывшемся файле укажите новое доменное имя (вставьте вместо `<NEW_DOMAIN_NAME>`) как значение ключей:

- `metadata → annotations → nginx.ingress.kubernetes.io/configuration-snippet → more_set_headers → Access-Control-Allow-Origin`;
- `spec → rules → host`.

Если на объекте Ingress установлен TLS-сертификат, также отредактируйте значение ключа `spec → tls → hosts`.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/configuration-snippet: |
      # Отредактируйте значение этого ключа.
      more_set_headers "Access-Control-Allow-Origin: https://<NEW_DOMAIN_NAME>";
# ...
spec:
  rules:
  # Отредактируйте значение этого ключа.
  - host: <NEW_DOMAIN_NAME>
  # ...
  tls:
  - hosts:
    # Если на объекте Ingress установлен TLS-сертификат, отредактируйте значение
этого ключа.
    - <NEW_DOMAIN_NAME>
    # ...
```

c. В пространстве имен `kube-prometheus-stack` находится объект Ingress `kube-prometheus-stack-grafana`, для которого аналогичным образом следует отредактировать значения ключей:

- `spec → rules → host`;
- `spec → tls → hosts` (если он существует).

```
kubectl -n kube-prometheus-stack edit ingress kube-prometheus-stack-grafana
```

Ключ `metadata → annotations → nginx.ingress.kubernetes.io/configuration-snippet → more_set_headers → Access-Control-Allow-Origin` для объекта `kube-prometheus-stack-grafana` отсутствует, добавлять его не нужно.

2. Отредактируйте значения секретов `evo.siem.global` и `evo.core.global`, используя команду:

```
kubectl -n <namespace> edit secret <secret>
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<secret>` — сначала `evo.siem.global`, затем `evo.core.global`.

  В обоих секретах нужно изменить значения ключей:

  - `MODULE_DOMAIN`;
  - `WEB_URL_HTTP`;
  - `WEB_URL_HTTPS`;
  - `WEB_URL_WS`.

▼ *Пример фрагмента файла секрета, который необходимо отредактировать*

```
apiVersion: v1
data:
  MODULE_DOMAIN: dGVzdF9kb21haW4=
  WEB_URL_HTTP: SFRUUDovL1dFQl9VUkxfSFRUUA==
  WEB_URL_HTTPS: SFRUUFM6Ly9XRUJfVVJMX0hUVFA=
  WEB_URL_WS: V1M6Ly9XRUJfVVJMX0hUVFA=
# ...
```

Значения секретов закодированы с помощью base64; новые значения также нужно указывать закодированными.

3. Перезагрузите деплойменты кластера, для которых менялись настройки Ingress:

```
kubectl -n <namespace> rollout restart deployment <deployment_name>
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<deployment_name>` — имя (`NAME`) перезагружаемого деплоймента:

- evo.core.bff;
- evo.core.host-app;
- evo.core.mf-feed;
- evo.siem.bff;
- evo.siem.documentation.

4. Перезагрузите деплоймент kube-prometheus-stack-grafana, находящийся в пространстве имен kube-prometheus-stack:

```
kubectl -n kube-prometheus-stack rollout restart deployment kube-prometheus-stack-grafana
```

5. Обновите конфигурацию сервиса evo.siem.frontend, выполнив следующие команды **на машине, на которой запускался инсталлятор системы**:

```
echo '{ "API_WS_URL": "wss://<NEW_DOMAIN_NAME>/api/graphql", "API_URL":
"https://<NEW_DOMAIN_NAME>/api/graphql", "NOT_ALLOWED_EXPERTISE_TYPES":
"calculation_schema,configuration_schema,domain,playbook,script" }' >
mf_config.json

helm upgrade -i evo.siem.frontend /opt/r-vision/common/helm/r-huc -n <namepsace>
--values /opt/r-vision/siem/helm/evo.siem.frontend/values.yaml --set-file
extraVars.CONFIG=mf_config.json --set
-json='global.ingress.hosts=["<NEW_DOMAIN_NAME>"]'
```

Здесь:

- <namespace> — имя пространства имен, в котором установлен кластер.
- <NEW_DOMAIN_NAME> — новое доменное имя.

Настройка системы

Перед началом работы с системой можно выполнить следующие настройки:

- Настройка интеграции с продуктами R-Vision
- Настройка формата даты
- Изменение пула адресов в MetalLB
- Настройка времени хранения запросов

Настройка интеграции с продуктами R-Vision

В данном разделе описана настройка интеграции экосистемы продуктов R-Vision.

Интеграция экосистемы продуктов R-Vision обеспечивает взаимодействие между различными продуктами R-Vision через веб-интерфейс системы. Поддерживается интеграция с R-Vision SOAR, R-

Vision Endpoint и R-Vision UEBA.

В системе продукты R-Vision SOAR и UEBA представлены в разделе **Инструменты** как **SOAR** и **UEBA**. Функциональность R-Vision Endpoint доступна в разделе **Ресурсы** как **Агенты**.

Для отображения внешних систем в веб-интерфейсе требуется настроить соединение, указав адреса подключения к этим системам в настройках сервера.

Настройки сервера уникальны для каждого тенанта системы.

Чтобы задать адреса для подключения к системам:

1. Откройте выпадающее меню учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Настройки сервера** (⚙). Откроется окно для настройки подключения.
3. Введите адреса серверов внешних систем в соответствующие поля:
   - SOAR URL (R-Vision SOAR);
   - Endpoint URL (R-Vision Endpoint);
   - UEBA URL (R-Vision UEBA).
4. Нажмите **Сохранить**.

   При успешном подключении разделы **SOAR** (R-Vision SOAR) и **UEBA** (R-Vision UEBA) станут доступны в меню **Инструменты**, а **Агенты** (R-Vision Endpoint) — в меню **Ресурсы**.

Разделы станут доступны после перезагрузки страницы или нового входа в систему.

Настройка формата даты

Администратор системы может настроить форматы дат, включая добавление новых пользовательских форматов, удаление существующих или редактирование текущих через переменную окружения в системе.

Работа с форматами даты

Для управления форматами дат в системе выполните следующие действия:

1. Подключитесь к кластеру, используя Lens.
2. Перейдите в раздел **Config** → **Config Maps**.
3. Найдите и выберите карту конфигурации **evo.core-host-app-env**.
4. В правой панели найдите поле переменной `DATE_FORMATS`.

Пример формата даты:

```
"[{\"label\":\"YYYY-MM-DDThh:mm:ss+hh:mm\", \"value\":\"yyyy-MM-
dd'T'HH:mm:ss.SSSZZ\"},{\"label\":\"DD.MM.YYYY HH:mm:ss\", \"value\":\"dd.MM.yyyy
HH:mm:ss\"},{\"label\":\"DD.MM.YYYY\", \"value\":\"dd.MM.yyyy\"}]"
```

где:

- `label` — описание формата, которое отображается в интерфейсе системы при выборе формата даты в настройках профиля пользователя.
- `value` — техническое представление формата даты и времени, соответствующее стандартам форматирования библиотеки Luxon.

Добавление нового формата даты

Чтобы добавить новый формат даты:

1. Добавьте его в JSON массив в поле `DATE_FORMATS`:

   ```
   "[..., {\"label\":\"Новый формат\", \"value\":\"новый-формат-даты\"}]"
   ```

2. Замените `"Новый формат"` и `"новый-формат-даты"` на желаемое описание и технический формат.
3. После внесения изменений сохраните конфигурацию.
4. Перейдите в раздел **Workloads** → **Deployments** и выполните перезагрузку сервиса **evo.core.host-app**, чтобы применить новые настройки формата даты.

Убедитесь, что изменения вступили в силу:

1. Перейдите к настройке формата дат и времени в учетной записи пользователя системы.
2. Убедитесь, что список доступных форматов дат и времени обновился согласно изменениям.

Редактирование формата даты

Чтобы изменить существующий формат даты:

1. Отредактируйте соответствующий объект в массиве, заменив значения `label` и `value` на новые.
2. После внесения изменений сохраните конфигурацию.
3. Перейдите в раздел **Workloads** → **Deployments** и выполните перезагрузку сервиса, чтобы применить новые настройки формата даты.

Удаление формата даты

Чтобы удалить формат даты:

1. Удалите соответствующий объект из массива. Убедитесь, что после удаления массив остается валидным JSON.
2. После внесения изменений сохраните конфигурацию.
3. Перейдите в раздел **Workloads** → **Deployments** и выполните перезагрузку сервиса, чтобы применить новые настройки формата даты.

Настройка параметра `label`

Параметр `label` в конфигурации формата даты служит для отображения описания формата в пользовательском интерфейсе, что позволяет пользователям выбрать нужный формат даты в

настройках своего профиля. Это описание должно быть понятным и точно отражать структуру данных, которую формат представляет.

Пример:

```
"[..., {\"label\":\"Полная дата и время\", \"value\":\"yyyy-MM-
dd'T'HH:mm:ss.SSSZZ\"}]"
```

где:

- `Полная дата и время` — описание формата даты, которое отображается в интерфейсе системы в настройках профиля пользователя при выборе формата даты.

Настройка параметра `value`

Параметр `value` в настройках формата даты определяет технический формат отображения дат и времени, который используется в системе. Формат указывается в соответствии с токенами форматирования, поддерживаемыми библиотекой Luxon.

Примеры настройки параметра `value` с расшифровкой токенов:

- Полная дата и время с часовым поясом:

  ```
  "yyyy-MM-dd'T'HH:mm:ss.SSSZZ"
  ```

  - `yyyy` — четырехзначный год.
  - `MM` — двузначный месяц.
  - `dd` — двузначный день.
  - `T` — разделитель даты и времени в международном стандарте ISO 8601.
  - `HH` — часы в 24-часовом формате.
  - `mm` — минуты.
  - `ss` — секунды.
  - `SSS` — миллисекунды.
  - `ZZ` — часовой пояс в формате +hh:mm или -hh:mm.

- Дата и время без часового пояса:

  ```
  "dd.MM.yyyy HH:mm:ss"
  ```

  - `dd` — двузначный день.
  - `MM` — двузначный месяц.
  - `yyyy` — четырехзначный год.
  - `HH` — часы в 24-часовом формате.

- `mm` — минуты.
- `ss` — секунды.

- Дата без времени:

```
"dd.MM.yyyy"
```

  - `dd` — двузначный день.
  - `MM` — двузначный месяц.
  - `yyyy` — четырехзначный год.

Изменение пула адресов в MetalLB

В состав дистрибутива системы включен балансировщик нагрузки MetalLB. В данном разделе описан процесс изменения пула адресов, который он выделяет сервисам кластера.

С помощью командной строки

Чтобы изменить пул адресов MetalLB с помощью командной строки:

1. Получите список пулов адресов кластера, выполнив следующую команду:

   ```
   kubectl get IPAddressPools -n metallb-system
   ```

2. Определите пул, который нужно изменить. В примере ниже префикс имени пула совпадает с именем пространства имен кластера (`evo`).

   ▼ *Пример вывода списка пулов адресов кластера*

   ```
   NAME        AGE
   evo-pool   146d
   ```

3. Откройте конфигурацию пула, выполнив следующую команду:

   ```
   kubectl edit IPAddressPools <pool_name> -n metallb-system
   ```

   Здесь:

   - `<pool_name>` — имя пула, полученное на шаге 2.

4. Отредактируйте ключ `spec → addresses` в конфигурационном файле, указав новый пул адресов:

   ```
   apiVersion: metallb.io/v1beta1
   kind: IPAddressPool
   metadata:
   # ...
   spec:
   ```

```
  # отредактируйте значение этого ключа
  addresses:
  - 10.0.0.15/32
  autoAssign: true
  avoidBuggyIPs: false
```

5. Перезагрузите деплоймент `metallb-controller`, чтобы применить изменения:

```
kubectl rollout restart deployment/metallb-controller
```

С помощью Lens

Чтобы изменить пул адресов MetalLB:

1. Подключитесь к кластеру с помощью Lens.
2. В выпадающем меню свойств кластера выберите пункт **Custom Resources** → **metallb.io** → **IP Address Pool**.
3. Выберите пространство имен `metallb-system` в выпадающем списке **Select Namespace** в правой части экрана. Отобразится таблица пулов IP-адресов.
4. Выберите пул адресов, который необходимо отредактировать.
5. Нажмите на кнопку **Edit** в правом верхнем углу экрана. В нижней части экрана отобразится файл конфигурации пула адресов.
6. Отредактируйте значение ключа `spec → addresses` аналогично п. 4 инструкции выше.
7. Перезагрузите деплоймент `metallb-controller`:
   a. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Deployments**.
   b. Выберите пространство имен `metallb-system` в выпадающем списке **Select Namespace** в правой части экрана. Отобразится список деплойментов.
   c. В строке таблицы, где находится деплоймент `metallb-controller`, нажмите на кнопку ⋮ в правом столбце и в выпадающем меню выберите пункт **Restart**.

Настройка времени хранения запросов

Срок хранения истории запросов можно настроить через переменную среды окружения в кластере Kubernetes.

Чтобы задать срок хранения истории:

1. Подключитесь к кластеру с помощью Lens.
2. В выпадающем меню свойств кластера выберите пункт **Config** → **Config Maps**.
3. Выберите карту конфигурации **evo.siem-query-log-manager-env** из списка.
4. В открывшейся панели справа задайте срок хранения истории запросов в поле переменной `SIEM_QUERY_LOG_MANAGER_QUERY_LOG_LIFETIME_INTERVAL`

   Формат:

```
SIEM_QUERY_LOG_MANAGER_QUERY_LOG_LIFETIME_INTERVAL: <number><unit>
```

Здесь:

- `<number>` — количество единиц.
- `<unit>` — единица измерения времени (например, `30d` для 30 дней).

5. Перейдите в **Workloads** → **Deployments** и перезагрузите сервис **evo.siem.query-log-manager**, чтобы изменения вступили в силу. Система начнет удалять историю с заданной периодичностью.

Убедитесь, что изменения вступили в силу:

1. Перейдите в **Workloads** → **Pods**.
2. Выберите под **evo.siem.query-log-manager** из списка.
3. В открывшейся панели убедитесь, что для переменной `SIEM_QUERY_LOG_MANAGER_QUERY_LOG_LIFETIME_INTERVAL` указан верный интервал.

Управление лицензией

Данный раздел описывает процесс работы с лицензией в системе. Работа осуществляется в разделе **Настройки** → **Лицензия** веб-интерфейса системы.

Раздел доступен только учетным записям со статусом **Суперадминистратор**.

О лицензии

Лицензия предоставляет доступ к функциям системы, а также определяет ресурсные ограничения и срок использования системы.

Система непрерывно отслеживает параметры лицензии и отображает уведомления при возникновении следующих ситуаций:

- Срок действия лицензии истекает в ближайшие 30 дней.
- Срок действия лицензии истек. Требуется активация нового ключа лицензии.
- Превышены ограничения хотя бы по одному параметру лицензии.

Соответствующие предупреждения также отображаются при просмотре сведений о лицензии.

Интерфейс раздела

Панель инструментов раздела включает кнопку **Активировать лицензию**, которая позволяет задать ключ лицензии в системе.

При наличии активированной лицензии в рабочей области раздела отображаются ее подробные сведения.

Работа с лицензией

Доступные операции над лицензией:

- Просмотр лицензии
- Активация лицензии

Просмотр лицензии

Чтобы просмотреть сведения о лицензии в системе, перейдите в раздел **Настройки** → **Лицензия**.

Если активированная лицензия отсутствует, в верхней части экрана отображается соответствующее предупреждение.

При наличии активированной лицензии в рабочей области раздела отображаются следующие сведения:

- **Статус** — текущий статус лицензии:
  - **Активна** — лицензия активна, и ее срок действия не истек.
  - **Истекает** — лицензия активна, и ее срок действия истекает в ближайшие 30 дней.

    При данном статусе в верхней части сведений о лицензии отображается предупреждение о скором истечении срока лицензии. Пользователь со статусом **Суперадминистратор** получит соответствующее уведомление при авторизации в системе.
  - **Истекла** — срок действия лицензии истек.

    При данном статусе в верхней части экрана отображается предупреждение об истечении срока лицензии и необходимости ее повторной активации. Пользователь со статусом **Суперадминистратор** получит соответствующее уведомление при авторизации в системе.
  - **Превышение параметров** — превышено хотя бы одно ограничение лицензии: EPS или входящий трафик.

    При данном статусе в верхней части экрана отображается предупреждение о превышении параметров лицензии. Пользователь со статусом **Суперадминистратор** получит соответствующее уведомление при авторизации в системе.
- **Server ID** — уникальный идентификатор сервера, для которого выдается лицензия.

  Справа от значения поля расположена кнопка, позволяющая скопировать идентификатор сервера в буфер обмена.
- **Срок действия лицензии** — даты начала и окончания действия лицензии.
- **Срок действия технической поддержки** — даты начала и окончания срока, в течение которого техническая поддержка R-Vision может оказывать помощь пользователям системы с текущей

лицензией.

- **EPS** — ограничение по суммарному количеству событий в секунду (EPS), которое может обрабатывать система с текущей лицензией.

  Если показатель EPS лицензией не ограничен, отображается надпись **Неограниченно**.

  Если система использует от 75% до 100% доступных ресурсов, показатель EPS выделяется желтым цветом.

  Если система использует более 100% доступных ресурсов, показатель EPS выделяется красным цветом, а в верхней части экрана отображается предупреждение о превышении параметров лицензии. Пользователь со статусом **Суперадминистратор** получит соответствующее уведомление при авторизации в системе.

- **Входящий трафик** — ограничение по суммарному объему входящего трафика в ГБ/сутки, который может обрабатывать система с текущей лицензией.

  Если показатель входящего трафика лицензией не ограничен, отображается надпись **Неограниченно**.

  Если система использует от 75% до 100% доступных ресурсов, показатель входящего трафика выделяется желтым цветом.

  Если система использует более 100% доступных ресурсов, показатель входящего трафика выделяется красным цветом, а в верхней части экрана отображается предупреждение о превышении параметров лицензии. Пользователь со статусом **Суперадминистратор** получит соответствующее уведомление при авторизации в системе.

Активация лицензии

Чтобы активировать лицензию:

1. Перейдите в раздел **Настройки** → **Лицензия**. Система отобразит сведения о текущей лицензии в системе.
2. Нажмите на кнопку **Активировать лицензию**. Откроется окно активации лицензии.
3. Нажмите на кнопку **Выбрать файл** (⊕) и выберите файл ключа лицензии в формате `.lic` на своем компьютере.

   При необходимости вы можете удалить загруженный файл с помощью кнопки 🗑.

4. Нажмите на кнопку **Активировать**. Система активирует лицензию и отобразит соответствующее уведомление.

Управление пространствами

Система включает в себя решение для управления кластерами продукта, а также оркестрации запускающихся на них компонентов и организации взаимодействия между ними.

Кластеры и пространства

Развертывание компонентов продукта в одном Kubernetes-кластере не всегда удовлетворяет бизнес-требованиям по работе с удаленным ресурсами, особенно если ресурсы расположены в DMZ-зоне. В таких случаях возникает необходимость развертывать компоненты в отдельных кластерах, которые могут быть размещены в разных сегментах сети. Система позволяет использовать такую топологию и управлять сервисами, которые запущены в этих кластерах.

Кластер Kubernetes, в котором разворачиваются все системные компоненты продукта, называется **основным кластером**. Часть компонентов продукта может быть развернута на дополнительных удаленных кластерах — **сателлитах**. В отличие от основного кластера, сателлит не обладает полным спектром функциональности продукта и может работать только с теми компонентами, которые на нем запущены.

Для подключения к различным кластерам продукта, а также развертывания и запуска его компонентов в этих кластерах используется специальная логическая среда — **пространство**. По умолчанию в системе всегда имеется **центральное пространство**, предназначенное для работы с компонентами в основном кластере.

Программные компоненты, разворачиваемые в пространствах, называются **сервисами**. Каждый сервис поставляется в виде Docker-образов при установке продукта и представляет собой набор контейнеров, приложений и их конфигураций.

В системе доступны следующие типы сервисов:

- **Коллектор** — сервис, который осуществляет сбор событий из источников, их обработку и дальнейшую отправку на хранение или во внешние системы.
- **Шлюз** — сервис, позволяющий управлять потоками данных между агентами на конечных устройствах и центральным кластером.

Таким образом, система предлагает функционал для управления пространствами в различных кластерах продукта и оркестрации его сервисов.

Работа с пространствами и сервисами осуществляется в разделе **Настройки → Пространства и сервисы** веб-интерфейса системы.

Конфигурация сервиса

- **Инфраструктурная конфигурация (Infra)** — содержит сведения о компонентах, которые должны быть развернуты в кластере. Infra-конфигурация не хранит в себе конфигурацию самих приложений, а только описывает необходимую для их запуска инфраструктуру: например, список контейнеров и переменные окружения.
- **Конфигурация приложений (Runtime)** — содержит сведения о поведении запускаемых приложений. Runtime-конфигурация описывает параметры, специфические для конкретного типа сервиса. Такие конфигурации создаются и предоставляются менеджерами конкретного типа сервисов. Runtime-конфигурация не является обязательной для работы сервиса и может отсутствовать.

Компоненты модуля управления пространствами

Модуль управления пространствами состоит из следующих компонентов:

1. **Менеджер пространств** — предназначен для создания пространств, подключения к ним и управления ими как объектами системы.
2. **Контроллер пространства** — разворачивается в каждом пространстве и предназначен для контроля за ресурсами пространства и за состоянием его сервисов:
   - **Контроль за ресурсами**: при попытке запуска сервиса контроллер определяет, достаточно ли в пространстве системных ресурсов для его развертывания. При достаточном объеме ресурсов контроллер запустит сервис.
   - **Контроль за сервисами**: контроллер регулярно отслеживает состояние сервисов, собирает логи об их изменении и следит, чтобы все запущенные сервисы работали. Так, например, если какой-нибудь сателлит будет перезапущен, контроллер заново развернет все необходимые сервисы на этом сателлите.
3. **Менеджер сервисов** — представляет собой конкретную реализацию менеджера определенного типа сервисов. Отвечает за формирование Runtime-конфигурации нужного типа сервиса и взаимодействие этого сервиса с системой.
4. **Шина** — предназначена для взаимодействия компонентов системы между собой посредством обмена командами и их результатами.

Работа с пространствами

Данный раздел описывает процесс работы с пространствами в системе. Работа осуществляется в разделе **Настройки** → **Пространства и сервисы** → **Пространства** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О пространствах

Пространство — это логическая среда в рамках кластера Kubernetes, предназначенная для развертывания и запуска сервисов в этом кластере.

Пространства могут быть созданы как в **основном кластере**, в котором развернуты все основные компоненты продукта, так и в **сателлитах** — удаленных кластерах, на которых развернута лишь часть компонентов.

По умолчанию в системе всегда имеется **центральное пространство**, предназначенное для работы с компонентами в основном кластере. Центральное пространство недоступно для изменения или удаления.

Для работы с пространствами используются следующие компоненты:

- **Менеджер пространств** — предназначен для создания пространств, подключения к ним и управления ими как объектами системы.
- **Контроллер пространства** — разворачивается в каждом пространстве и предназначен для

контроля за ресурсами пространства и за состоянием его сервисов:

- **Контроль за ресурсами**: при попытке запуска сервиса контроллер определяет, достаточно ли в пространстве системных ресурсов для его развертывания. При достаточном объеме ресурсов контроллер запустит сервис.
- **Контроль за сервисами**: контроллер регулярно отслеживает состояние сервисов, собирает логи об их изменении и следит, чтобы все запущенные сервисы работали. Так, например, если какой-нибудь сателлит будет перезапущен, контроллер заново развернет все необходимые сервисы на этом сателлите.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать подключение к новому пространству.
- Поле **Поиск** предназначено для быстрого поиска подключений к пространствам в отображаемом списке по названию, IP-адресу или домену.

В рабочей области отображается таблица имеющихся подключений к пространствам. Таблица содержит следующие столбцы:

- **Название** — название подключения к пространству, используемое для его идентификации в системе.
- **Статус** — текущий статус подключения к пространству: **Активный**, **Ошибка** или **Нет соединения**.
- **IP/Домен** — IP-адрес или домен пространства.

При работе с таблицей подключений к пространствам доступны следующие операции:

- Поиск подключений по полям **Название** и **IP/Домен**.
- Сортировка подключений по полям **Название** и **IP/Домен**.

При выборе конкретного подключения к пространству в правой части рабочей области отображается его карточка с детальной информацией.

В верхней части карточки приведено название подключения, а также имеются кнопки для изменения ( ) и удаления ( ) подключения к пространству.

Карточка подключения содержит сведения, сгруппированные по следующим секциям:

- **Статистика** — метрики ресурсов, занимаемых пространством.
- **Информация** — основная информация о подключении к пространству.
- **Сервисы** — список сервисов в пространстве.

Работа с подключением к пространству

Доступные операции над подключениями к пространствам:

- Создание подключения к пространству
- Просмотр подключения к пространству

- Изменение подключения к пространству
- Удаление подключения к пространству

Создание подключения к пространству

Чтобы создать подключение к пространству:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**. Система отобразит сведения об имеющихся подключениях к пространствам.
2. Нажмите на кнопку **Создать** (┼). Отобразится окно создания подключения.
3. Введите уникальный идентификатор подключения.
4. Введите название подключения.
5. Введите IP-адрес или домен, по которому расположено пространство.

Домен сателлита должен быть уникальным в рамках системы. Создание нескольких подключений к одному пространству с одинаковым доменом недоступно.

6. Введите порт для подключения к пространству. Допустимые значения: 1—65535.
7. При необходимости введите описание подключения к пространству.
8. Нажмите на кнопку **Создать**. Система создаст подключение к пространству и отобразит соответствующее уведомление. Подключение отобразится в списке раздела **Настройки** → **Пространства и сервисы** → **Пространства**.

Просмотр подключения к пространству

Чтобы просмотреть подключение к пространству:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**. Система отобразит сведения об имеющихся подключениях к пространствам.
2. Нажмите на строку подключения в списке. Система отобразит в правой части экрана карточку этого подключения с подробной информацией о нем.

В верхней части карточки отображается название подключения к пространству.

Карточка подключения содержит сведения, сгруппированные по следующим секциям:

- **Статистика** — метрики ресурсов, занимаемых пространством:
  - **CPU** — количество ядер и уровень загруженности процессора в пространстве.
  - **RAM** — объем и уровень загруженности оперативной памяти в пространстве.
- **Информация** — основная информация о подключении к пространству:
  - **ID** — уникальный идентификационный код подключения, генерируемый автоматически при его создании.
  - **Название** — имя подключения, используемое для его идентификации в системе.
  - **IP-адрес/домен** — IP-адрес или домен кластера, в котором расположено пространство.

- **Статус** — текущий статус подключения: **Активный**, **Ошибка** или **Нет соединения**.
- **Порт** — порт кластера, в котором расположено пространство.
- **Тенант** — название тенанта, к которому относится подключение.
- **Описание** — краткое описание функциональности пространства.
- **Дата создания** — дата и время создания подключения.
- **Создал** — пользователь, инициировавший создание подключения.
- **Дата изменения** — дата и время последнего изменения подключения.
- **Изменил** — пользователь, выполнивший последнее изменение подключения.

- **Сервисы** — список сервисов в пространстве. Список представлен в виде таблицы со следующими столбцами:
  - **Активен** — переключатель состояния сервиса.
  - **Название** — название сервиса.
  - **CPU** — уровень загруженности процессора сервиса. При наведении курсора на данный столбец отображаются полные сведения о количестве ядер и уровне загруженности процессора для текущего сервиса.
  - **RAM** — уровень загруженности оперативной памяти сервиса. При наведении курсора на данный столбец отображаются полные сведения об объеме и уровне загруженности оперативной памяти для текущего сервиса.

  В правой части строки сервиса в таблице имеется кнопка →, по которой осуществляется переход в раздел **Настройки → Пространства и сервисы → Сервисы** и открывается карточка выбранного сервиса.

  Над таблицей имеется кнопка **Добавить** (+) для добавления нового сервиса.

Изменение подключения к пространству

Изменение доступно только для пространств без запущенных сервисов.

Чтобы изменить подключение к пространству:

1. Перейдите в раздел **Настройки → Пространства и сервисы → Пространства**. Система отобразит сведения об имеющихся подключениях к пространствам.
2. Нажмите на строку подключения в списке. Система отобразит в правой части экрана карточку этого подключения с подробной информацией о нем.
3. Нажмите на кнопку в верхней части карточки подключения. Отобразится окно настроек подключения к пространству.
4. Внесите изменения в требуемые поля.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке и списке подключений.

Удаление подключения к пространству

Удаление недоступно для центрального пространства.

- При удалении подключения осуществляется только разрыв соединения с пространством, само пространство физически не удаляется и впоследствии может быть использовано повторно для создания нового подключения.
- Если в пространстве имеются запущенные сервисы, то при удалении подключения эти сервисы станут недоступны для работы или удаления. Чтобы восстановить доступ к сервисам, необходимо заново создать подключение к их пространству.

Чтобы удалить подключение к пространству:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**. Система отобразит сведения об имеющихся подключениях к пространствам.
2. Нажмите на строку подключения в списке. Система отобразит в правой части экрана карточку этого подключения с подробной информацией о нем.
3. Нажмите на кнопку в верхней части карточки подключения. Отобразится окно подтверждения действия.
4. Нажмите на кнопку **Удалить**. Система удалит подключение к пространству и отобразит соответствующее уведомление. Удаленное подключение будет исключено из списка раздела.

Работа с сервисами

Данный раздел описывает процесс работы с сервисами в системе. Работа осуществляется в разделе **Настройки** → **Пространства и сервисы** → **Сервисы** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О сервисах

Сервис — это программный компонент, который может разворачиваться в продукте. Сервис создается и запускается в рамках пространства — специальной логической среды, осуществляющей подключение к какому-либо кластеру продукта.

Каждый сервис поставляется в виде Docker-образов при установке продукта и представляет собой набор контейнеров, приложений и их конфигураций.

В системе доступны следующие типы сервисов:

- **Коллектор** — сервис, который осуществляет сбор событий из источников, их обработку и дальнейшую отправку на хранение или во внешние системы.
- **Шлюз** — сервис, позволяющий управлять потоками данных между агентами на конечных устройствах и центральным кластером.

Конфигурация сервиса:

- **Инфраструктурная конфигурация (Infra)** — содержит сведения о компонентах, которые должны быть развернуты в кластере. Infra-конфигурация не хранит в себе конфигурацию самих приложений, а только описывает необходимую для их запуска инфраструктуру: например, список контейнеров и переменные окружения.
- **Конфигурация приложений (Runtime)** — содержит сведения о поведении запускаемых приложений. Runtime-конфигурация описывает параметры, специфические для конкретного типа сервиса. Такие конфигурации создаются и предоставляются менеджерами конкретного типа сервисов. Runtime-конфигурация не является обязательной для работы сервиса и может отсутствовать.

Для работы с сервисами используются следующие компоненты:

- **Контроллер пространства** — разворачивается в каждом пространстве и предназначен для контроля за ресурсами пространства и за состоянием его сервисов:
  - **Контроль за ресурсами**: при попытке запуска сервиса контроллер определяет, достаточно ли в пространстве системных ресурсов для его развертывания. При достаточном объеме ресурсов контроллер запустит сервис.
  - **Контроль за сервисами**: контроллер регулярно отслеживает состояние сервисов, собирает логи об их изменении и следит, чтобы все запущенные сервисы работали. Так, например, если какой-нибудь сателлит будет перезапущен, контроллер заново развернет все необходимые сервисы на этом сателлите.
- **Менеджер сервисов** — представляет собой конкретную реализацию менеджера определенного типа сервисов. Отвечает за формирование Runtime-конфигурации нужного типа сервиса и взаимодействие этого сервиса с системой.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (＋) позволяет создать новый сервис.
- Поле **Поиск** предназначено для быстрого поиска сервисов в отображаемом списке по названию или пространству.
- Выпадающий список **Тип** позволяет настроить отображение списка сервисов в соответствии с выбранным типом сервиса.

В рабочей области отображается таблица имеющихся сервисов. Таблица содержит следующие столбцы:

- **Активен** — переключатель статуса сервиса.
- **Название** — название сервиса, используемое для его идентификации в системе.
- **Тип** — тип сервиса.
- **Пространство** — пространство, в котором создан сервис.

При работе с таблицей сервисов доступны следующие операции:

- Поиск сервисов по полям **Название** и **Пространство**.
- Фильтрация сервисов по полю **Тип**. Тип выбирается из выпадающего списка.
- Сортировка сервисов по полям **Название** и **Пространство**.

При выборе конкретного сервиса в правой части рабочей области отображается его карточка с детальной информацией.

Работа с сервисом

Доступные операции над сервисами:

- Создание сервиса
- Просмотр сервиса
- Удаление сервиса
- Запуск сервиса
- Остановка сервиса

Создание сервиса

Чтобы создать сервис:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Нажмите на кнопку **Создать** (＋) и выберите тип сервиса. Откроется окно создания сервиса.
3. Задайте настройки сервиса.

   Настройки сервисов идентичны их настройкам в разделах системы, в которых они используются:

   - коллекторы — в разделе **Ресурсы** → **Коллекторы**;
   - шлюзы — в разделе **Агенты** → **Шлюзы**.
4. Нажмите на кнопку **Создать**. Система создаст сервис и отобразит соответствующее уведомление. Сервис отобразится в списке раздела **Настройки** → **Пространства и сервисы** → **Сервисы**, а также в списке сервисов в карточке соответствующего подключения к пространству.

   Новый сервис типа **Коллектор** также отобразится в списке раздела **Ресурсы** → **Коллекторы**, а новый сервис типа **Шлюз** — в списке раздела **Агенты** → **Шлюзы**.

Вы также можете создать сервис из карточки подключения к пространству. Для этого:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**. Система отобразит сведения об имеющихся подключениях к пространствам.
2. Нажмите на строку подключения в списке. Система отобразит в правой части экрана карточку этого подключения с подробной информацией о нем.
3. Нажмите на кнопку **Добавить** (＋) над списком сервисов в карточке подключения и выберите тип сервиса **Коллектор**. Откроется окно создания

сервиса.

4. Задайте настройки сервиса.
5. Нажмите на кнопку **Создать**. Система создаст сервис и отобразит соответствующее уведомление. Сервис отобразится в списке раздела **Настройки** → **Пространства и сервисы** → **Сервисы**, а также в списке сервисов в карточке соответствующего подключения к пространству.

Просмотр сервиса

Чтобы просмотреть сервис:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Нажмите на строку сервиса в списке. Система отобразит в правой части экрана карточку этого сервиса с подробной информацией о нем.

Вы также можете открыть карточку сервиса из карточки подключения к пространству:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**. Система отобразит сведения об имеющихся подключениях к пространствам.
2. Нажмите на строку подключения в списке. Система отобразит в правой части экрана карточку этого подключения с подробной информацией о нем.
3. Нажмите на кнопку → в правой части строки нужного сервиса в списке сервисов. Система осуществит переход в раздел **Настройки** → **Пространства и сервисы** → **Сервисы** и откроет карточку выбранного сервиса.

В верхней части карточки отображается название сервиса.

В карточке расположен переключатель состояния сервиса, а также отображаются метрики ресурсов, занимаемых сервисом:

- **CPU** — количество ядер и уровень загруженности процессора в сервисе.
- **RAM** — объем и уровень загруженности оперативной памяти в сервисе.

Карточка также содержит следующие поля:

- **ID** — уникальный идентификационный код сервиса, генерируемый автоматически при его создании.
- **Название** — имя сервиса, используемое для его идентификации в системе.
- **Тип** — тип сервиса.
- **Статус** — текущий статус сервиса:
  - **Неустановленный**;
  - **Ожидание**;
  - **Запускается**;

- **Запущен**;
- **Останавливается**;
- **Остановлен**;
- **Ошибка**;
- **Неизвестная ошибка**.

В строке поля также отображается кнопка , по которой можно инициировать обновление статуса.

- **Пространство** — пространство, в котором создан сервис.
- **Тенант** — название тенанта, к которому относится сервис.
- **Дата создания** — дата и время создания сервиса.
- **Создал** — пользователь, инициировавший создание сервиса.
- **Дата изменения** — дата и время последнего изменения сервиса.
- **Изменил** — пользователь, выполнивший последнее изменение сервиса.
- **Статус конфигурации** — статус Runtime-конфигурации статуса: **Сохранен**, **Применен** или **Ошибка**.

В нижней части карточки также содержится кнопка **Посмотреть логи**, нажатие на которую позволяет просмотреть список логов сервиса.

Удаление сервиса

Чтобы удалить сервис:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Нажмите на строку сервиса в списке. Система отобразит в правой части экрана карточку этого сервиса с подробной информацией о нем.
3. Нажмите на кнопку в верхней части карточки сервиса. Отобразится окно подтверждения действия.
4. Нажмите на кнопку **Удалить**. Система удалит выбранный сервис и отобразит соответствующее уведомление. Удаленный сервис будет исключен из списка раздела, а также из списка сервисов в карточке соответствующего подключения к пространству.

Запуск сервиса

Для того чтобы сервис мог начать работу, его нужно запустить.

Чтобы запустить сервис:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Нажмите на строку сервиса в списке. Система отобразит в правой части экрана карточку этого сервиса с подробной информацией о нем.
3. Убедитесь, что в карточке сервиса открыта вкладка **Информация**.

4. Переведите переключатель в верхней части карточки в активное положение и подтвердите запуск сервиса. Система запустит сервис и отобразит соответствующее уведомление.

Вы также можете запустить сервис из списка сервисов. Для этого:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Переведите переключатель слева от строки сервиса в списке в активное положение и подтвердите запуск сервиса. Система запустит сервис и отобразит соответствующее уведомление.

Запуск сервиса также возможен через карточку подключения к пространству, в котором создан сервис. Для этого:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**. Система отобразит сведения об имеющихся подключениях к пространствам.
2. Нажмите на строку подключения в списке. Система отобразит в правой части экрана карточку этого подключения с подробной информацией о нем.
3. Переведите переключатель слева от строки сервиса в карточке подключения в активное положение и подтвердите запуск сервиса. Система запустит сервис и отобразит соответствующее уведомление.

Остановка сервиса

Чтобы остановить работающий сервис:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Нажмите на строку сервиса в списке. Система отобразит в правой части экрана карточку этого сервиса с подробной информацией о нем.
3. Убедитесь, что в карточке сервиса открыта вкладка **Информация**.
4. Переведите переключатель в верхней части карточки в неактивное положение и подтвердите остановку сервиса. Система остановит сервис и отобразит соответствующее уведомление.

Вы также можете остановить сервис через список сервисов. Для этого:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Сервисы**. Система отобразит сведения об имеющихся сервисах.
2. Переведите переключатель слева от строки сервиса в списке в неактивное положение и подтвердите остановку сервиса. Система остановит сервис и отобразит соответствующее уведомление.

Остановка сервиса также возможна через карточку подключения к пространству, в котором запущен сервис. Для этого:

1. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**. Система отобразит сведения об имеющихся подключениях к пространствам.

2. Нажмите на строку подключения в списке. Система отобразит в правой части экрана карточку этого подключения с подробной информацией о нем.
3. Переведите переключатель слева от строки сервиса в карточке подключения в неактивное положение и подтвердите остановку сервиса. Система остановит сервис и отобразит соответствующее уведомление.

Управление доступом

Данный раздел посвящен управлению тенантами и учетными записями пользователей в системе. В подразделах описаны ключевые аспекты и действия, доступные администраторам для работы с тенантами, учетными записями, политиками безопасности, ролями и настройками доступа.

Содержание раздела:

- Тенанты
- Пользователи
- Группы пользователей
- Роли
  - Стандартные роли
- Политики входа и пароля
  - Настройка времени жизни для токенов доступа
- LDAP-соединения
  - Основные пользовательские атрибуты Active Directory
- Политики управления доступом к событиям

Тенанты

Данный раздел описывает процесс работы с тенантами в системе. Работа осуществляется в разделе **Настройки** → **Управление доступом** → **Тенанты** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О тенантах

Система поддерживает режим мультитенантности, при котором один экземпляр системы в инфраструктуре организации (далее — **родительский**, или **основной**, тенант) позволяет изолированным частям системы (далее — **дочерние** тенанты) получать, хранить и обрабатывать события независимо друг от друга.

В системе можно работать только в рамках одного тенанта одновременно. Текущий тенант отображается в левом верхнем углу веб-интерфейса под логотипом продукта. В пределах тенанта можно настраивать систему для отдельных групп пользователей, регулируя их права доступа к разделам, доступные операции и доступ к данным хранилища, оставаясь в пределах одной

инфраструктуры. События тенантов хранятся отдельно.

В системе доступны следующие типы тенантов:

- **Основной тенант** — тенант, устанавливаемый в системе по умолчанию. Имеет доступ к данным всех дочерних тенантов. Основной тенант нельзя удалить или изменить.
- **Дочерний тенант** — изолированная часть основной инфраструктуры системы, которая содержит ресурсы и данные, выделенные из основного тенанта. Он предназначен для предоставления ограниченного доступа к этим ресурсам и настройкам только определенной группе пользователей. Каждый дочерний тенант работает независимо, обеспечивая доступ и управление только собственными ресурсами и данными.

Доступ к тенантам регулируется в настройках пользователей. Пользователь может иметь доступ к нескольким тенантам одновременно. В каждом тенанте ему могут быть назначены разные роли. Роль состоит из разрешений и действует в указанных тенантах. Если роль не привязана к конкретному тенанту, она доступна во всех тенантах. Один и тот же пользователь может иметь разные роли в разных тенантах.

При формировании API-запроса к системе, работа происходит в том тенанте, который указан в заголовке запроса.

Наследование тенанта

Ресурсы и события системы доступны только в том тенанте, к которому они относятся.

Объекту автоматически присваивается идентификатор тенанта, в рамках которого он был создан. Этот идентификатор указывается в поле `tenantId` объекта. После создания объекта его тенант невозможно изменить.

Текущий тенант отображается в левом верхнем углу веб-интерфейса системы под логотипом программы.

Например, тенант корреляционного события наследуется от создавшего его коллектора. Коллектор получает идентификатор того тенанта, в котором он был создан.

Интерфейс раздела

Панель инструментов раздела включает следующие компоненты:

- Кнопка **Добавить** (+) позволяет создать новый тенант.
- Поле **Поиск** предназначено для быстрого поиска тенантов в отображаемом списке по названию.

В рабочей области отображается таблица имеющихся в системе тенантов, которые доступны пользователю. Таблица содержит следующие столбцы:

- **Активен** — текущий статус тенанта: включен или выключен.
- **Имя** — имя тенанта, используемое для его идентификации в системе.
- **Описание** — описание тенанта.

При работе с таблицей тенантов доступны следующие операции:

- Поиск тенантов по полю **Имя**.
- Сортировка тенантов по полю **Имя**.

При выборе конкретного тенанта в правой части рабочей области отображается его карточка с детальной информацией.

Работа с тенантами

Доступные операции над тенантами:

- Добавление тенанта
- Просмотр тенанта
- Включение тенанта
- Отключение тенанта
- Изменение тенанта
- Переключение между тенантами

Добавление тенанта

Чтобы добавить тенант:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Тенанты**. Система отобразит сведения о доступных тенантах.
2. Нажмите на кнопку **Добавить** ( ). Отобразится окно добавления тенанта.
3. Введите название тенанта.
4. При необходимости введите описание тенанта.
5. Нажмите на кнопку **Добавить**. Система создаст тенант и отобразит соответствующее уведомление. Новый тенант появится в списке раздела.

Просмотр тенанта

Чтобы просмотреть тенант:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Тенанты**. Система отобразит сведения о доступных тенантах.
2. Нажмите на строку тенанта в списке. Система отобразит в правой части экрана карточку этого тенанта с подробной информацией о нем.

В верхней части карточки отображается название тенанта. Справа от названия расположена кнопка , по которой открывается выпадающее меню с действиями, доступными для выбранного тенанта.

Карточка содержит основные сведения о тенанте:

- **ID** — уникальный идентификационный код тенанта, генерируемый автоматически при его создании.
- **Родительский тенант** — основной тенант, к которому принадлежит текущий тенант.
- **Имя** — имя тенанта, используемое для его идентификации в системе.

- **Статус** — текущий статус тенанта: включен или выключен.
- **Описание** — описание тенанта.
- **Дата создания** — дата и время создания тенанта.
- **Создал** — пользователь, инициировавший создание тенанта.
- **Дата изменения** — дата и время последнего изменения тенанта.
- **Изменил** — пользователь, выполнивший последнее изменение тенанта.

В верхней части карточки также расположен переключатель статуса тенанта.

Включение тенанта

Включение тенанта делает его доступным для выбора в списке доступных тенантов системы.

Чтобы включить тенант:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Тенанты**. Система отобразит сведения о доступных тенантах.
2. Переведите переключатель состояния в столбце **Активен** в строке нужного тенанта в активное положение. Система включит тенант и отобразит соответствующее уведомление. Тенант станет доступным для выбора.

Вы также можете включить тенант через его карточку. Для этого:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Тенанты**. Система отобразит сведения о доступных тенантах.
2. Нажмите на строку тенанта в списке. Система отобразит в правой части экрана карточку этого тенанта с подробной информацией о нем.
3. Переведите переключатель состояния тенанта в верхней части карточки в активное положение. Система включит тенант и отобразит соответствующее уведомление. Тенант станет доступным для выбора.

Отключение тенанта

Отключение тенанта делает его недоступным для выбора в списке доступных тенантов системы.

Чтобы отключить тенант:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Тенанты**. Система отобразит сведения о доступных тенантах.
2. Переведите переключатель состояния в столбце **Активен** в строке нужного тенанта в неактивное положение. Система отобразит окно подтверждения действия.
3. Нажмите на кнопку **Отключить**. Система отключит тенант и отобразит соответствующее уведомление. Тенант перестанет быть доступным для выбора.

Вы также можете отключить тенант через его карточку. Для этого:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Тенанты**. Система

отобразит сведения о доступных тенантах.

2. Нажмите на строку тенанта в списке. Система отобразит в правой части экрана карточку этого тенанта с подробной информацией о нем.
3. Переведите переключатель в верхней части тенанта в неактивное положение. Отобразится окно подтверждения действия.
4. Нажмите на кнопку **Отключить**. Система отключит тенант и отобразит соответствующее уведомление. Тенант перестанет быть доступным для выбора.

Изменение тенанта

Чтобы изменить тенант:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Тенанты**. Система отобразит сведения о доступных тенантах.
2. Нажмите на строку тенанта в списке. Система отобразит в правой части экрана карточку этого тенанта с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек тенанта.
4. Внесите требуемые изменения в конфигурацию тенанта.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке тенанта и в списке раздела.

Переключение между тенантами

Одновременно можно работать только в рамках одного тенанта. Если пользователь имеет доступ к нескольким тенантам, он может выбрать, данные какого тенанта будут отображаться в веб-интерфейсе системы.

Чтобы выбрать тенант:

1. На главной панели веб-интерфейса под логотипом продукта нажмите на название тенанта. Система отобразит список доступных тенантов.
2. Выберите тенант, данные которого необходимо отобразить в веб-интерфейсе системы. В разделах веб-интерфейса отобразятся только данные и аналитика, которые относятся к выбранному тенанту.

Пользователи

Данный раздел описывает процесс работы с учетными записями пользователей в системе. Работа осуществляется в разделе **Настройки** → **Управление доступом** → **Пользователи** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О пользователях

Учетная запись пользователя — это средство идентификации и аутентификации лица в системе, включающее в себя уникальные учетные данные, такие как логин и пароль, а также набор присвоенных ролей, которые определяют доступные для пользователя разделы интерфейса и операции в системе. Метаданные учетной записи могут содержать информацию о ФИО, должности и дополнительное описание.

В системе можно создавать учетные записи следующих типов:

- **Локальный пользователь** — учетная запись для доступа и работы пользователей в системе.
- **Пользователь из домена** — учетная запись, созданная через интеграцию с LDAP-сервером и предназначенная для управления пользователями из внешнего каталога.
- **Бот** — учетная запись, предназначенная для автоматизированного взаимодействия с системой через API-запросы. При создании учетной записи типа **Бот** ей присваивается API-токен.

Роли и разрешения пользователей

Пользователю могут быть назначены одна или несколько ролей. Роли могут быть назначены как непосредственно учетной записи, так и через группы пользователей, в которых состоит пользователь. В разных тенантах один и тот же пользователь может иметь разные роли. Разрешения из всех ролей пользователя в рамках одного тенанта суммируются. При конфликте разрешений применяется наибольшее из доступных. Пользователь может иметь одну или несколько ролей, однако наличие хотя бы одной роли является обязательным.

Предусмотрена возможность создания пользовательских ролей или использования стандартных.

В системе по умолчанию установлены следующие стандартные роли со своим набором разрешений:

- **Администратор** — отвечает за настройку и поддержку инфраструктуры системы, обновление конфигураций и обеспечение безопасности данных. Имеет полный доступ к административным функциям. Обладает всеми возможными разрешениями в системе, имеет полный контроль над всеми тенантами системы.

  При создании учетной записи пользователя, ей можно присвоить статус **Суперадминистратор**. Это предоставит все доступные в системе разрешения.

- **Инженер** — занимается настройкой операционных процессов системы, разработкой решений для оптимизации этих процессов и их внедрением. Отвечает за создание и установку конфигураций конвейера, настройку коллектора. Действия инженера ограничены рамками определенных тенантов.
- **Пользователь** — работает с данными системы в рамках определенных тенантов, выполняет анализ и визуализацию данных для поддержки управленческих решений. Имеет ограниченный доступ к ресурсам системы.

Учетную запись с разрешениями администратора может создать и редактировать только другой администратор. Из системы нельзя удалить последнюю учетную запись со статусом **Суперадминистратор**.

Интерфейс раздела

Панель инструментов раздела включает следующие компоненты:

- Кнопка **Добавить** (+) позволяет добавить нового пользователя.
- Поле **Поиск** предназначено для быстрого поиска пользователей в отображаемом списке по полям **ФИО** и **Логин**.
- Выпадающий список **Роль** позволяет отфильтровать список пользователей по ролям.
- Выпадающий список **Тип** позволяет отфильтровать список пользователей по типам учетной записи.

В рабочей области отображается таблица имеющихся в системе учетных записей пользователей. Таблица содержит следующие столбцы:

- **Логин** — логин пользователя в системе.
- **Статус** — текущий статус пользователя: активен или заблокирован.
- **Тип пользователя** — тип учетной записи пользователя.
- **ФИО** — полные фамилия, имя и отчество пользователя.
- **Роли** — перечень ролей, назначенных пользователю.

При работе с таблицей учетных записей доступны следующие операции:

- Поиск пользователей по полям **Логин** и **ФИО**.
- Фильтрация пользователей по полям **Роль** и **Тип пользователя**. Роли и типы выбираются из выпадающих списков.
- Сортировка пользователей по полям **Логин**, **Тип пользователя** и **ФИО**.

При выборе конкретной учетной записи в правой части рабочей области отображается ее карточка с детальной информацией.

Карточка учетной записи содержит следующие вкладки:

- **Основное** — основная информация об учетной записи пользователя.
- **Роли** — список ролей, назначенных пользователю.
- **Разрешения** — список разрешений пользователя.

Работа с пользователем

Доступные операции над пользователем:

- Добавление пользователя
- Просмотр пользователя
- Блокировка пользователя
- Активация пользователя
- Изменение пользователя

- Удаление пользователя

Добавление пользователя

Вы можете добавить учетную запись пользователя с нужным набором ролей и параметров.

В системе можно создавать учетные записи следующих типов:

- **Локальный пользователь** — учетная запись для доступа и работы пользователей в системе.
- **Пользователь из домена** — учетная запись, созданная через интеграцию с LDAP-сервером и предназначенная для управления пользователями из внешнего каталога.
- **Бот** — учетная запись, предназначенная для автоматизированного взаимодействия с системой через API-запросы. При создании учетной записи типа **Бот** ей присваивается API-токен.

Добавление локального пользователя

Чтобы добавить локального пользователя:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на кнопку **Добавить** (+) и выберите из выпадающего списка тип **Локальный пользователь**. Система отобразит окно добавления локального пользователя.
3. Заполните следующие поля:
   - **Логин** — задайте уникальный логин для учетной записи пользователя в системе.
   - **Пароль** — установите пароль для входа пользователя в систему. Пароль должен удовлетворять заданным требованиям.
   - **Группа пользователей** (опционально) — выберите из выпадающего списка группы, в которые должен входить пользователь.
   - **ФИО** (опционально) — укажите полное имя пользователя.
   - **Должность** — укажите должность пользователя в организации.
   - **Номер телефона** (опционально) — укажите номер телефона пользователя.
   - **Email** (опционально) — укажите адрес электронной почты пользователя.
   - **Описание** (опционально) — добавьте краткое описание учетной записи пользователя.
4. Назначьте роли пользователю, чтобы определить перечень его разрешений в системе.

   Чтобы добавить роль:

   a. Нажмите на кнопку **Добавить роль** (+). Появится строка настройки роли.

   b. Выберите из выпадающего списка роль, которую необходимо назначить пользователю.

   c. При необходимости выберите из выпадающего списка тенанты, в рамках которых должна действовать роль.

   > Если тенанты не выбраны, роль действует во всех тенантах.

   Чтобы удалить добавленную роль, нажмите на кнопку в строке настройки роли.

Если активирован переключатель **Суперадминистратор**, учетной записи предоставляются все доступные в системе разрешения.

5. Нажмите на кнопку **Добавить**. Система создаст учетную запись пользователя и отобразит соответствующее уведомление. Новая учетная запись появится в списке раздела.

Добавление пользователя из домена

Пользователь из домена — учетная запись, созданная через интеграцию с LDAP-сервером, для управления пользователями из внешнего каталога. Синхронизируются следующие поля учетных записей: логин, ФИО, должность, статус, электронная почта, номер телефона.

Чтобы добавить пользователя из домена:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на кнопку **Добавить** (＋) и выберите из выпадающего списка тип **Пользователь из домена**. Система отобразит окно добавления доменных пользователей.
3. Выберите одно из доступных LDAP-соединений из выпадающего списка **Домен**. Система отобразит каталоги выбранного LDAP-соединения в виде дерева в поле **Каталоги**.
4. Выберите необходимый каталог из дерева. Система отобразит информацию о пользователях из выбранного каталога в поле **Пользователи**.

   Для быстрого поиска каталога введите запрос в строку поиска над деревом каталогов.

По умолчанию поиск выполняется с начала названия каталога. Например, если необходимо найти каталог с названием "domain users", то его можно найти по запросу `domain`. По запросу `users` данный каталог найден не будет.

Чтобы выполнить поиск по подстроке, используйте следующий синтаксис: `*подстрока*`. Например, для поиска каталога с названием "domain users" по подстроке "users" используйте запрос `*users*`.

5. Выберите одного или нескольких пользователей, для которых необходимо создать учетную запись. Можно выбрать до 10 пользователей из одного или нескольких каталогов.

   Для быстрого поиска пользователей введите запрос в строку поиска над списком пользователей. В списке отобразятся пользователи, в именах которых встречается введенный запрос.

6. При необходимости выберите из выпадающего списка группы, в которые должны входить доменные пользователи.
7. Нажмите на кнопку **Далее**. Отобразится вкладка для настройки разрешений пользователей.
8. Назначьте роли доменным пользователям, чтобы определить перечень их разрешений в системе.

   Чтобы добавить роль:

   a. Нажмите на кнопку **Добавить роль** (＋). Появится строка настройки роли.

b. Выберите из выпадающего списка роль, которую необходимо назначить пользователю.

c. При необходимости выберите из выпадающего списка тенанты, в рамках которых должна действовать роль.

Если тенанты не выбраны, роль действует во всех тенантах.

Чтобы удалить добавленную роль, нажмите на кнопку в строке настройки роли.

Если активирован переключатель **Суперадминистратор**, учетной записи предоставляются все доступные в системе разрешения.

9. Нажмите на кнопку **Создать**. Система создаст учетные записи для выбранных пользователей и отобразит соответствующее уведомление. При создании учетных записей будут синхронизированы следующие поля: логин, ФИО, должность, статус, email, номер телефона. Новые учетные записи появятся в списке раздела.

Если в систему добавлены пользователи с одинаковыми логинами, но из разных доменов, то при входе им нужно дополнительно указывать домен в поле логина.

Добавление бота

Бот — учетная запись, предназначенная для автоматизированного взаимодействия с системой через API-запросы. При создании учетной записи типа **Бот** ей автоматически присваивается API-токен.

Чтобы создать бота:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на кнопку **Добавить** (+) и выберите из выпадающего списка тип **Бот**. Система отобразит окно добавления бота.
3. Заполните следующие поля:
   - **Логин** — задайте уникальный логин для учетной записи пользователя в системе.
   - **Группа пользователей** (опционально) — выберите из выпадающего списка группы, в которые должен входить пользователь.
   - **Описание** (опционально) — добавьте краткое описание учетной записи пользователя.
4. Назначьте роли пользователю, чтобы определить перечень его разрешений в системе.

   Чтобы добавить роль:

   a. Нажмите на кнопку **Добавить роль** (+). Появится строка настройки роли.

   b. Выберите из выпадающего списка роль, которую необходимо назначить пользователю.

   c. При необходимости выберите из выпадающего списка тенанты, в рамках которых должна действовать роль.

Если тенанты не выбраны, роль действует во всех тенантах.

Чтобы удалить добавленную роль, нажмите на кнопку в строке настройки роли.

Если активирован переключатель **Суперадминистратор**, учетной записи предоставляются все доступные в системе разрешения.

5. Нажмите на кнопку **Добавить**. Система создаст бота и отобразит соответствующее уведомление. Новый бот появится в списке раздела.

При создании бота система автоматически генерирует API-токен, который используется для авторизации в API-запросах.

Просмотр пользователя

Чтобы просмотреть пользователя:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на строку пользователя в списке. Система отобразит в правой части экрана карточку этого пользователя с подробной информацией о нем.

В верхней части карточки отображается логин учетной записи. Справа от логина расположена кнопка , по которой открывается выпадающее меню с действиями, доступными для выбранной учетной записи.

Карточка учетной записи содержит следующие вкладки:

- **Основное** — основная информация об учетной записи пользователя.
- **Роли** — список ролей, назначенных пользователю.
- **Разрешения** — список разрешений пользователя.

Для учетной записи типа **Бот** в нижней части карточки отображаются следующие кнопки:

- **Скопировать токен** () — копирует токен в буфер обмена.
- **Перевыпустить токен** () — обновляет API-токен бота, делая предыдущий токен недействительным.

Вкладка Основное

На вкладке **Основное** отображается основная информация об учетной записи. Вкладка содержит следующие поля:

- **Логин** — уникальный идентификатор пользователя в системе.
- **Статус** — текущий статус пользователя: активен или заблокирован.
- **Тип** — тип учетной записи пользователя: локальный, доменный или бот.
- **Описание** — краткое описание учетной записи пользователя.
- **Дата создания** — дата и время создания учетной записи.
- **Создал** — пользователь, инициировавший создание учетной записи.

- **Дата изменения** — дата и время последнего изменения учетной записи.
- **Изменил** — пользователь, выполнивший последнее изменение учетной записи.
- **ФИО** — полное имя пользователя.
- **Должность** — должность пользователя в организации.
- **Суперадминистратор** — указывает, обладает ли пользователь статусом **Суперадминистратор**.
- **Номер телефона** — номер телефона пользователя.
- **Email** — адрес электронной почты пользователя.

В верхней части вкладки также содержится переключатель статуса пользователя.

Вкладка Роли

Для пользователей со статусом **Суперадминистратор** эта вкладка не отображается.

На вкладке **Роли** приведен перечень ролей, назначенных пользователю.

По умолчанию для каждой роли отображается только ее название. Чтобы получить более подробные сведения о роли, нажмите на кнопку ⌄ в правой части ее строки. Для роли отобразятся следующие сведения:

- **Источники** — список источников роли:
  - Если роль была добавлена вручную при создании учетной записи, в качестве источника отображается **Собственная роль**.
  - Если роль была наследована от группы, в которой содержится учетная запись, в качестве источника отображается название этой группы.
- **Тенанты** — список тенантов, в которых действует роль.

Вкладка Разрешения

Для пользователей со статусом **Суперадминистратор** эта вкладка не отображается.

На вкладке **Разрешения** приведен полный перечень разрешений, которыми обладает пользователь.

Список разрешений представлен в виде таблицы со следующими столбцами:

- **Разрешение** — название разрешения.
- **Раздел** — раздел системы, который затрагивается разрешением. Путь до раздела указывается через /.
- **Модуль** — функциональный модуль платформы, к которому относится разрешение.
- **Тенанты и организации** — список тенантов, в рамках которых выдано разрешение.

При работе с таблицей разрешений доступны следующие операции:

- Поиск разрешений по всем полям.

- Сортировка разрешений по полям **Разрешение** и **Модуль**.

Блокировка пользователя

При необходимости вы можете заблокировать учетную запись пользователя, чтобы запретить ему авторизоваться и работать с системой.

Чтобы заблокировать учетную запись пользователя:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на строку пользователя в списке. Система отобразит в правой части экрана карточку этого пользователя с подробной информацией о нем.
3. Убедитесь, что в карточке открыта вкладка **Основное**.
4. В верхней части карточки установите переключатель статуса в неактивное положение. Система заблокирует пользователя и отобразит соответствующее уведомление.

Активация пользователя

Активация заблокированной учетной записи возобновляет возможность авторизоваться и работать с системой через эту учетную запись.

Заблокированные учетные записи могут быть активированы только администраторами, пользователями с соответствующими разрешениями в составе своих ролей и учетными записями со статусом **Суперадминистратор**.

Чтобы активировать учетную запись пользователя:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на строку пользователя в списке. Система отобразит в правой части экрана карточку этого пользователя с подробной информацией о нем.
3. Убедитесь, что в карточке открыта вкладка **Основное**.
4. В верхней части карточки установите переключатель статуса в активное положение. Система активирует пользователя и отобразит соответствующее уведомление.

Изменение пользователя

Чтобы изменить данные пользователя:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на строку пользователя в списке. Система отобразит в правой части экрана карточку этого пользователя с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек пользователя.
4. Внесите требуемые изменения в конфигурацию пользователя.

5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке пользователя и в списке раздела.

При изменении списка ролей пользователя доступные ему разделы меню обновятся после перезагрузки страницы или повторного входа в систему.

Удаление пользователя

Чтобы удалить пользователя:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Наведите курсор на строку пользователя в списке и нажмите на появившуюся кнопку в правой части строки. Отобразится окно подтверждения действия.
3. Нажмите на кнопку **Удалить**. Система удалит пользователя и отобразит соответствующее уведомление. Удаленный пользователь будет исключен из списка раздела.

Удалить пользователя можно также из его карточки. Для этого:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на строку пользователя в списке. Система отобразит в правой части экрана карточку этого пользователя с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения действия.
4. Нажмите на кнопку **Удалить**. Система удалит пользователя и отобразит соответствующее уведомление. Удаленный пользователь будет исключен из списка раздела.

Вы также можете удалить сразу несколько пользователей. Для этого:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Установите флажки напротив тех пользователей, которых вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно подтверждения действия.
4. Нажмите на кнопку **Удалить**. Система удалит выбранных пользователей и отобразит соответствующее уведомление. Удаленные пользователи будут исключены из списка раздела.

Группы пользователей

Данный раздел описывает процесс работы с группами пользователей в системе. Работа осуществляется в разделе **Настройки** → **Управление доступом** → **Группы пользователей** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют

соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О группах пользователей

Группа пользователей объединяет учетные записи для назначения общих прав управления системой. В системе поддерживаются следующие типы групп пользователей:

- **Локальная группа** — группа, куда могут быть включены любые учетные записи в системе. Ее состав регулируется вручную через веб-интерфейс системы.
- **Доменная группа** — группа, связанная с каталогом LDAP. Ее состав и данные учетных записей автоматически синхронизируются с указанным LDAP-каталогом.

Группам пользователей могут быть назначены роли, которые определяют права пользователей группы в системе. Одной группе можно присвоить сразу несколько ролей.

Группе не может быть присвоен статус **Суперадминистратор** — этот статус может быть присвоен только индивидуальным пользователям.

Назначение ролей в группах пользователей является необязательным. Группу можно создать и сохранить без ролей, в этом случае пользователи группы не получат дополнительных прав. Группы пользователей не привязаны к тенантам системы.

Интерфейс раздела

Панель инструментов раздела включает следующие компоненты:

- Кнопка **Добавить** (+) позволяет добавить новую группу пользователей.
- Выпадающий список **Тип** позволяет отфильтровать список групп пользователей по типу.
- Поле **Поиск** предназначено для быстрого поиска пользователей в отображаемом списке по полям **Название** и **Ключ**.

В рабочей области отображается таблица имеющихся в системе групп пользователей. Таблица содержит следующие столбцы:

- **Ключ** — уникальный идентификатор группы.
- **Название** — наименование группы пользователей.
- **Тип группы** — тип группы: локальная или доменная.
- **Описание** — описание группы.
- **Дата создания** — дата и время создания группы.
- **Дата изменения** — дата и время последнего изменения группы.

При работе с таблицей групп пользователей доступны следующие операции:

- Поиск групп пользователей по полям **Название** и **Ключ**.
- Фильтрация групп пользователей по полю **Тип группы**. Тип группы выбирается из выпадающего списка.

- Сортировка групп пользователей по полям **Ключ**, **Название**, **Дата создания** и **Дата изменения**.

При выборе конкретной группы в правой части рабочей области отображается ее карточка с детальной информацией.

Карточка группы пользователей содержит следующие вкладки:

- **Основное** — основная информация о группе.
- **Состав группы** — список пользователей, входящих в группу.
- **Роли** — список ролей, назначенных группе.
- **Разрешения** — список разрешений группы.

Работа с группой пользователей

Доступные операции над группой пользователей:

- Создание группы пользователей
- Просмотр группы пользователей
- Изменение группы пользователей
- Изменение состава локальной группы пользователей
- Синхронизация группы пользователей
- Удаление группы пользователей

Создание группы пользователей

В системе можно создавать группы пользователей следующих типов:

- **Локальная группа** — группа, куда могут быть включены любые пользователи системы.
- **Доменная группа** — группа, связанная с каталогом LDAP.

Создание локальной группы пользователей

При необходимости вы можете сохранить группу без пользователей.

Чтобы создать локальную группу пользователей:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит сведения о доступных группах.
2. Нажмите на кнопку **Создать** (+) и выберите из выпадающего списка тип **Локальная группа**. Система отобразит окно добавления локальной группы.
3. Заполните следующие поля:
   - **Название** — задайте уникальное название для группы пользователей.
   - **Ключ** — задайте уникальный идентификатор для группы. Ключ может содержать цифры (1–9), латинские буквы (a–z) и специальные символы (-_).
   - **Описание** (опционально) — добавьте краткое описание группы.

4. При необходимости назначьте роли группе пользователей, чтобы определить перечень их разрешений в системе.

   Чтобы добавить роль:

   a. Нажмите на кнопку **Добавить роль** (+). Появится строка настройки роли.

   b. Выберите из выпадающего списка роль, которую необходимо назначить всем пользователям в группе.

   c. При необходимости выберите из выпадающего списка тенанты, в рамках которых должна действовать роль.

   Если тенанты не выбраны, роль действует во всех тенантах.

   Чтобы удалить добавленную роль, нажмите на кнопку в строке настройки роли.

5. При необходимости в поле **Состав группы** установите флажки слева от пользователей, которые должны быть включены в группу.

   Список пользователей представлен в виде таблицы со следующими столбцами:

   - **Логин** — логин пользователя в системе.
   - **ФИО** — полное имя пользователя.

   При работе с таблицей пользователей доступны следующие операции:

   - Поиск пользователей по всем полям.
   - Сортировка пользователей по всем полям.

   Чтобы отобразить в таблице только уже выбранных пользователей, переведите переключатель **Только выбранное** над таблицей в активное положение.

6. Нажмите на кнопку **Создать**. Система создаст локальную группу пользователей и отобразит соответствующее уведомление. Новая группа появится в списке раздела.

Создание доменной группы пользователей

Чтобы создать доменную группу пользователей:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит сведения о доступных группах.
2. Нажмите на кнопку **Создать** (+) и выберите из выпадающего списка тип **Доменная группа**. Система отобразит окно добавления локальной группы.
3. Заполните следующие поля:
   - **Название** — задайте уникальное название для группы пользователей.
   - **Ключ** — задайте уникальный идентификатор для группы. Ключ может содержать цифры (1–9), латинские буквы (a–z) и специальные символы (-_).
   - **Описание** (опционально) — добавьте краткое описание группы.

4. Выберите одно из доступных LDAP-соединений из выпадающего списка. Система отобразит каталоги и группы выбранного LDAP-соединения в виде дерева в поле **Группы**.
5. Выберите необходимую группу пользователей из дерева. Система отобразит список входящих в группу пользователей в поле **Пользователи**.

   Для быстрого поиска каталога или группы введите запрос в строку поиска над деревом каталогов.

   По умолчанию поиск выполняется с начала названия каталога или группы. Например, если необходимо найти каталог с названием "domain users", то его можно найти по запросу `domain`. По запросу `users` данный каталог найден не будет.

   Чтобы выполнить поиск по подстроке, используйте следующий синтаксис: `*подстрока*`. Например, для поиска каталога с названием "domain users" по подстроке "users" используйте запрос `*users*`.
6. Нажмите на кнопку **Далее**. Отобразится вкладка для настройки разрешений в группе.
7. Назначьте роли доменным пользователям, чтобы определить перечень их разрешений в системе.

   Чтобы добавить роль:

   a. Нажмите на кнопку **Добавить роль** (+). Появится строка настройки роли.

   b. Выберите из выпадающего списка роль, которую необходимо назначить всем пользователям в группе.

   c. При необходимости выберите из выпадающего списка тенанты, в рамках которых должна действовать роль.

   Если тенанты не выбраны, роль действует во всех тенантах.

   Чтобы удалить добавленную роль, нажмите на кнопку в строке настройки роли.
8. При необходимости в поле **Состав группы** установите флажки слева от пользователей, которые должны быть включены в группу.

   Список пользователей представлен в виде таблицы со следующими столбцами:

   - **Логин** — логин пользователя в системе.
   - **ФИО** — полное имя пользователя.

   При работе с таблицей пользователей доступны следующие операции:

   - Поиск пользователей по всем полям.
   - Сортировка пользователей по всем полям.

   Чтобы отобразить в таблице только уже выбранных пользователей, переведите переключатель **Только выбранное** над таблицей в активное положение.

9. Нажмите на кнопку **Создать**. Система создаст доменную группу пользователей и отобразит соответствующее уведомление. Новая группа появится в списке раздела.

После создания группы система выполнит синхронизацию, в ходе которой состав группы обновится в соответствии с указанными параметрами.

Просмотр группы пользователей

Чтобы просмотреть группу пользователей:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит список доступных групп.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы с подробной информацией о ней.

В верхней части карточки отображается название группы. Справа от названия расположены кнопки действий, доступных для выбранной группы.

Карточка группы пользователей содержит следующие вкладки:

- **Основное** — основная информация о группе.
- **Состав группы** — список пользователей, входящих в группу.
- **Роли** — список ролей, назначенных группе.
- **Разрешения** — список разрешений группы.

Для доменной группы в нижней части карточки также содержится кнопка **Синхронизировать** (🔄) для синхронизации группы.

Вкладка Основное

На вкладке **Основное** отображается основная информация о группе пользователей. Вкладка содержит следующие поля:

- **Название** — имя группы.
- **Ключ** — уникальный идентификатор группы в системе.
- **Тип группы** — тип группы: локальная или доменная.
- **Описание** — краткое описание группы.
- **Дата создания** — дата и время создания группы.
- **Создал** — пользователь, инициировавший создание группы.
- **Дата изменения** — дата и время последнего изменения группы.
- **Изменил** — пользователь, выполнивший последнее изменение группы.
- **LDAP-домен** — имя LDAP-домена. Поле отображается только для доменных групп.
- **Название в LDAP** — название группы в LDAP-каталоге. Поле отображается только для доменных групп.
- **Синхронизировано** — дата и время последней синхронизации с LDAP. Поле отображается только для доменных групп.

Вкладка Состав группы

На вкладке **Состав группы** отображается список пользователей, входящих в группу.

Список пользователей представлен в виде таблицы со следующими столбцами:

- **Логин** — логин пользователя.
- **ФИО** — полное имя пользователя.

В таблице доступна сортировка пользователей по всем полям.

Для локальных групп на вкладке доступна опция изменения состава группы через кнопки и . Для доменных групп вкладка **Состав группы** отображает список пользователей без возможности его изменения. Состав доменной группы изменяется автоматически при синхронизации с LDAP-каталогом.

Вкладка Роли

На вкладке **Роли** карточки группы пользователей отображается список ролей, которые назначаются пользователям в группе.

По умолчанию для каждой роли отображается только ее название. Чтобы получить более подробные сведения о роли, нажмите на кнопку в правой части ее строки. Для роли отобразится список тенантов, в которых она действует.

Вкладка Разрешения

На вкладке **Разрешения** карточки группы пользователей приведен полный перечень разрешений, которыми обладают пользователи в группе.

Список разрешений представлен в виде таблицы со следующими столбцами:

- **Разрешение** — название разрешения.
- **Раздел** — раздел системы, который затрагивается разрешением. Путь до раздела указывается через /.
- **Модуль** — функциональный модуль платформы, к которому относится разрешение.
- **Тенанты и организации** — список тенантов, в рамках которых выдано разрешение.

При работе с таблицей разрешений доступны следующие операции:

- Поиск разрешений по всем полям.
- Сортировка разрешений по полям **Разрешение** и **Модуль**.

Изменение группы пользователей

Чтобы изменить группу пользователей:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит список доступных групп.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы с подробной информацией о ней.

3. Нажмите на кнопку в верхнем правом углу карточки группы. Отобразится окно настройки группы.
4. Измените необходимые параметры.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке группы и в списке раздела.

Изменение состава локальной группы пользователей

Состав группы можно изменить только в локальной группе. Для доменной группы пользователи добавляются или удаляются автоматически при синхронизации с LDAP-каталогом.

Вы можете изменить состав локальной группы, добавив в нее новых пользователей или удалив из нее существующих.

Чтобы изменить состав локальной группы пользователей:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит список доступных групп.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы с подробной информацией о ней.
3. Перейдите на вкладку **Состав группы**.
4. Нажмите на кнопку **Изменить** () в верхней части вкладки. Откроется окно изменения состава группы.

   В окне отображается список доступных пользователей в системе. Список представлен в виде таблицы со следующими столбцами:

   - **Логин** — логин пользователя.
   - **ФИО** — полное имя пользователя.

   При работе с таблицей пользователей доступны следующие операции:

   - Поиск пользователей по всем полям.
   - Сортировка пользователей по всем полям.

   Чтобы добавить нового пользователя в группу, установите флажок слева от его строки.

   Чтобы удалить существующего пользователя из группы, уберите флажок слева от его строки.

   Чтобы отобразить в таблице только уже выбранных пользователей, переведите переключатель **Только выбранное** над таблицей в активное положение.

5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке группы.

Удалять пользователей из группы также можно через таблицу на вкладке **Состав группы**. Для этого:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит список доступных групп.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы с подробной информацией о ней.
3. Перейдите на вкладку **Состав группы**.
4. Наведите курсор на строку пользователя, которого необходимо удалить, и нажмите на появившуюся кнопку в правой части строки. Система удалит пользователя из группы и отобразит соответствующее уведомление.

Синхронизация группы пользователей

Синхронизация доступна только для доменных групп пользователей. Изменение состава локальных групп выполняется вручную.

Синхронизация обновляет состав группы пользователей в системе на основе данных из LDAP-сервера. Синхронизируются следующие поля учетных записей: логин, ФИО, должность, статус, электронная почта, номер телефона.

Запуск ручной синхронизации данных группы пользователей с LDAP-сервером осуществляется через кнопку **Синхронизировать** в карточке группы пользователей.

Чтобы синхронизировать данные доменной группы:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит список доступных групп.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы с подробной информацией о ней.
3. Нажмите на кнопку **Синхронизировать** () в нижней части карточки. Начнется синхронизация данных группы с LDAP-сервером.

   По завершении операции система отобразит уведомление о статусе синхронизации.

Удаление группы пользователей

Чтобы удалить группу:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Группы пользователей**. Система отобразит список доступных групп.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы с подробной информацией о ней.
3. Нажмите на кнопку в верхнем правом углу карточки группы. Отобразится окно подтверждения удаления группы.
4. Нажмите на кнопку **Удалить**. Система удалит группу пользователей и отобразит соответствующее уведомление. Удаленная группа будет исключена из списка раздела.

Роли

Данный раздел описывает процесс работы с ролями пользователей в системе. Работа осуществляется в разделе **Настройки** → **Управление доступом** → **Роли** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О ролях

Роль — это набор разрешений, который определяет доступные для пользователя разделы интерфейса и операции в системе.

Разрешение — это право выполнения определенного действия или наличие доступа к определенной функциональности системы. Разрешения в системе разделены по модулям.

Роли могут быть назначены пользователям индивидуально или через группы пользователей. Пользователь может иметь одну или несколько ролей одновременно. Если роль не привязана к конкретному тенанту, она доступна во всех тенантах системы. В разных тенантах один и тот же пользователь может иметь разные роли. Разрешения из всех ролей пользователя в рамках одного тенанта суммируются. При конфликте разрешений применяется наибольшее из доступных. Пользователь может иметь одну или несколько ролей, однако наличие хотя бы одной роли является обязательным.

Система поддерживает создание пользовательских ролей с произвольными наборами разрешений.

Стандартные роли

В системе по умолчанию установлены следующие стандартные роли с предопределенными наборами разрешений:

- **Администратор** — отвечает за настройку и поддержку инфраструктуры системы, обновление конфигураций и обеспечение безопасности данных. Имеет полный доступ к административным функциям. Обладает всеми возможными разрешениями в системе, имеет полный контроль над всеми тенантами системы.

При создании учетной записи пользователя, ей можно присвоить статус **Суперадминистратор**. Это предоставит все доступные в системе разрешения.

- **Инженер** — занимается настройкой операционных процессов системы, разработкой решений для оптимизации этих процессов и их внедрением. Отвечает за создание и установку конфигураций конвейера, настройку коллектора. Действия инженера ограничены рамками определенных тенантов.
- **Пользователь** — работает с данными системы в рамках определенных тенантов, выполняет анализ и визуализацию данных для поддержки управленческих решений. Имеет ограниченный доступ к ресурсам системы.

Суперадминистраторы

Пользователи со статусом **Суперадминистратор** имеют высший уровень доступа и могут выполнять любые операции в системе, включая создание и управление другими ролями и пользователями.

Интерфейс раздела

Панель инструментов раздела включает следующие компоненты:

- Кнопка **Создать** (＋) позволяет создать новую роль.
- Поле **Поиск** предназначено для быстрого поиска ролей в отображаемом списке по названию.

В рабочей области отображается таблица имеющихся в системе ролей, которые доступны пользователю. Таблица содержит следующие столбцы:

- **Имя** — имя роли, используемое для ее идентификации в системе.
- **Тип** — тип роли: стандартная или пользовательская.
- **Описание** — описание роли.

При работе с таблицей ролей доступны следующие операции:

- Поиск ролей по полю **Имя**.
- Сортировка ролей по полям **Имя** и **Тип**.

При выборе конкретной роли в правой части рабочей области отображается ее карточка с детальной информацией.

Карточка роли содержит следующие вкладки:

- **Разрешения** — список разрешений роли.
- **Информация** — основная информация о роли.

Работа с ролью

Доступные операции над ролью:

- Создание роли
- Просмотр роли
- Изменение роли
- Создание копии роли
- Удаление роли

Создание роли

Вы можете создать роль с нужным набором разрешений.

Чтобы создать роль:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Роли**. Система отобразит сведения о

доступных ролях.

2. Нажмите на кнопку **Создать** (┼). Система отобразит окно создания роли.
3. Введите название роли.
4. При необходимости введите описание роли.
5. В поле **Разрешения** установите флажки слева от разрешений, которыми должна обладать роль. Разрешения представлены в виде дерева и сгруппированы по модулям платформы и разделам системы.

Если у разрешения есть связанные разрешения, необходимые для его корректной работы, при его выборе они выдаются автоматически. При этом связанные разрешения становятся недоступными для отключения.

Так, если выдать разрешение на управление сущностью системы, автоматически будет выдано разрешение и на ее просмотр. Например, при выборе разрешения **Управление секретами** дополнительно выдается разрешение **Просмотр секретов**.

Чтобы выдать для роли сразу все разрешения из конкретного модуля или раздела, установите флажок слева от его заголовка в дереве. При этом, если в будущих версиях продукта в рамках выбранного модуля или раздела появятся новые разрешения, они будут автоматически выданы для данной роли.

Для быстрого поиска разрешений введите запрос в строку поиска над деревом разрешений. В дереве отобразятся только те разрешения, в названии которых содержится введенный запрос.

Чтобы отобразить в дереве только уже выбранные разрешения, переведите переключатель **Только выбранное** над деревом разрешений в активное положение.

6. Нажмите на кнопку **Создать**. Система создаст роль и отобразит соответствующее уведомление. Новая роль появится в списке раздела.

Просмотр роли

Чтобы просмотреть роль:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Роли**. Система отобразит сведения о доступных ролях.
2. Нажмите на строку роли в списке. Система отобразит в правой части экрана карточку этой роли с подробной информацией о ней.

В верхней части карточки отображается название роли. Справа от названия расположена кнопка ⋮, по которой открывается выпадающее меню с действиями, доступными для выбранной роли.

Карточка роли содержит следующие вкладки:

- **Разрешения** — список разрешений роли.
- **Информация** — основная информация о роли.

Вкладка Разрешения

На вкладке **Разрешения** приведен полный перечень разрешений, назначенных роли.

Список разрешений представлен в виде таблицы со следующими столбцами:

- **Разрешение** — название разрешения.
- **Раздел** — раздел системы, который затрагивается разрешением. Путь до раздела указывается через /.
- **Модуль** — функциональный модуль платформы, к которому относится разрешение.

При работе с таблицей разрешений доступны следующие операции:

- Поиск разрешений по всем полям.
- Сортировка разрешений по полям **Разрешение** и **Модуль**.

Вкладка Информация

На вкладке **Информация** отображается основная информация о роли. Вкладка содержит следующие поля:

- **Имя** — имя роли, используемое для ее идентификации в системе.
- **Тип** — тип роли: стандартная или пользовательская.
- **Описание** — описание роли.
- **Дата создания** — дата и время создания роли.
- **Создал** — пользователь, инициировавший создание роли.
- **Дата изменения** — дата и время последнего изменения роли.
- **Изменил** — пользователь, выполнивший последнее изменение роли.

Изменение роли

Чтобы изменить роль:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Роли**. Система отобразит сведения о доступных ролях.
2. Нажмите на строку роли в списке. Система отобразит в правой части экрана карточку этой роли с подробной информацией о ней.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек роли.
4. Внесите требуемые изменения в конфигурацию роли.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке роли и в списке раздела.

При изменении разрешений роли, разделы меню, доступные пользователям с этой ролью, обновятся после перезагрузки страницы или повторного входа в систему.

Создание копии роли

Чтобы создать копию роли:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Роли**. Система отобразит сведения о доступных ролях.
2. Наведите курсор на строку роли в списке и нажмите на появившуюся кнопку в правой части строки. Отобразится окно создания копии выбранной роли.
3. Внесите требуемые изменения в конфигурацию роли.
4. Нажмите на кнопку **Сохранить**. Система создаст копию выбранной роли и отобразит соответствующее уведомление. Новая роль появится в списке раздела.

Вы также можете создать копию роли из ее карточки. Для этого:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Роли**. Система отобразит сведения о доступных ролях.
2. Нажмите на строку роли в списке. Система отобразит в правой части экрана карточку этой роли с подробной информацией о ней.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Создать копию**. Отобразится окно создания копии выбранной роли.
4. Внесите требуемые изменения в конфигурацию роли.
5. Нажмите на кнопку **Сохранить**. Система создаст копию выбранной роли и отобразит соответствующее уведомление. Новая роль появится в списке раздела.

Удаление роли

Чтобы удалить роль:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Роли**. Система отобразит сведения о доступных ролях.
2. Наведите курсор на строку роли в списке и нажмите на появившуюся кнопку в правой части строки. Отобразится окно подтверждения действия.
3. Нажмите на кнопку **Удалить**. Система удалит роль и отобразит соответствующее уведомление. Удаленная роль будет исключена из списка раздела.

Вы также можете удалить роль из ее карточки. Для этого:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Роли**. Система отобразит сведения о доступных ролях.
2. Нажмите на строку роли в списке. Система отобразит в правой части экрана карточку этой роли с подробной информацией о ней.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения действия.
4. Нажмите на кнопку **Удалить**. Система удалит роль и отобразит соответствующее уведомление. Удаленная роль будет исключена из списка

раздела.

Стандартные роли

О стандартных ролях

Роль — это набор разрешений, который определяет доступные для пользователя разделы интерфейса и операции в системе.

Стандартные роли — это роли с предопределенным набором разрешений. Стандартные роли не могут быть изменены или удалены.

В системе предустановлены следующие стандартные роли:

- **Администратор** — отвечает за настройку и поддержку инфраструктуры системы, обновление конфигураций и обеспечение безопасности данных. Имеет полный доступ к административным функциям. Обладает всеми возможными разрешениями в системе, имеет полный контроль над всеми тенантами системы.

При создании учетной записи пользователя, ей можно присвоить статус **Суперадминистратор**. Это предоставит все доступные в системе разрешения.

- **Инженер** — занимается настройкой операционных процессов системы, разработкой решений для оптимизации этих процессов и их внедрением. Отвечает за создание и установку конфигураций конвейера, настройку коллектора. Действия инженера ограничены рамками определенных тенантов.
- **Пользователь** — работает с данными системы в рамках определенных тенантов, выполняет анализ и визуализацию данных для поддержки управленческих решений. Имеет ограниченный доступ к ресурсам системы.

Разрешения стандартных ролей пользователей

Разрешения в системе сгруппированы по следующим секциям:

- Системные разрешения
- Разрешения для работы с менеджером агентов
- Разрешения для работы с событиями ИБ
- Разрешения для работы с пространствами
- Разрешения для работы с уязвимостями
- Разрешения для работы с доменами
- Разрешения для работы с автоматизацией

Системные разрешения

Системные разрешения сгруппированы в разделе **Настройки** дерева разрешений. Они определяют, может ли пользователь управлять доступом других пользователей к системе, а также работать с LDAP-соединениями.

Системными разрешениями обладает только стандартная роль **Администратор**.

Группа разрешений	Разрешения
Управление доступом	• Политики пароля и входа ◦ Изменение политик пароля и входа ◦ Просмотр политик пароля и входа
	• Роли ◦ Создание ролей ◦ Удаление ролей ◦ Политики ▪ Создание политик ▪ Удаление политик ▪ Изменение политик ▪ Просмотр политик ◦ Изменение ролей ◦ Просмотр ролей
	• Тенанты ◦ Создание тенантов ◦ Изменение тенантов ◦ Просмотр тенантов
	• Группы пользователей ◦ Создание групп пользователей ◦ Удаление групп пользователей ◦ Изменение групп пользователей ◦ Просмотр групп пользователей
	• Пользователи ◦ Создание пользователей ◦ Удаление пользователей ◦ Изменение пользователей ◦ Просмотр пользователей
LDAP-соединения	• Создание LDAP-соединений • Удаление LDAP-соединений • Запуск синхронизации LDAP-соединения • Изменение LDAP-соединений • Просмотр LDAP-соединений

Разрешения для работы с менеджером агентов

Разрешения для работы с менеджером агентов сгруппированы в разделе **Агенты** дерева разрешений. Они определяют, может ли пользователь устанавливать и настраивать агенты, а также управлять работой агентов, групп агентов и шлюзов.

Разрешениями для работы с менеджером агентов обладает только стандартная роль **Администратор**.

Группа разрешений	Разрешения
Настройка конфигурации группы агентов	• Изменение конфигурации • Просмотр информации о конфигурации
Группы агентов	• Создание/дублирование группы • Удаление группы • Редактирование группы • Получение информации о группах
Агенты	• Назначение группы агенту • Удаление агента • Выполнение действия на агенте (все действия) • Удаление группы у агента • Получение информации об агентах
Шлюзы	• Создание шлюза • Удаление шлюза • Запуск шлюза • Остановка шлюза • Редактирование шлюза • Просмотр информации о шлюзах
Настройки	• Агенты

Разрешения для работы с событиями ИБ

Разрешения для работы с событиями ИБ сгруппированы в разделе **SIEM** дерева разрешений. Они определяют, доступны ли для пользователя такие разделы, как **Ресурсы** и **Экспертиза**.

Всеми разрешениями для работы с событиями ИБ обладает только стандартная роль **Администратор**.

Группа разрешений	Разрешения	Инженер	Пользователь
Дашборды	Просмотр дашбордов	✓	✓
	Управление дашбордами	✓	✓
Экспертиза	Просмотр элементов экспертизы	✓	✓
	Управление элементами экспертизы	✓	
Отчеты	Просмотр отчетов	✓	✓
	Управление отчетами	✓	✓
Ресурсы → Активные списки	Просмотр активных списков	✓	✓
	Управление активными списками	✓	
Ресурсы → Коллекторы	Просмотр коллекторов	✓	✓
	Управление коллекторами	✓	
Ресурсы → Базы данных событий	Просмотр БД событий	✓	✓
	Управление БД событий	✓	
Ресурсы → Модели событий	Просмотр моделей событий	✓	✓
	Управление моделями событий	✓	
Ресурсы → Хранилища событий	Просмотр хранилищ событий	✓	✓
	Управление хранилищами событий	✓	
Ресурсы → Секреты	Просмотр секретов	✓	
	Управление секретами	✓	
Поиск		✓	✓
Настройки сервера	Просмотр настроек сервера	✓	✓
	Изменение настроек сервера	✓	
Инструменты → Политики управления доступа к событиям	Просмотр политик управления доступом к событиям	✓	
	Управление политиками доступом к событиям	✓	
Инструменты → Мониторинг		✓	✓
Инструменты → Аудит источников	Просмотр аудита источников	✓	✓
	Управление аудитом источников	✓	
Инструменты → Диспетчер запросов	Просмотр записей диспетчера запросов	✓	✓
	Управление записями диспетчера запросов	✓	✓
	Просмотр записей ресурсов пользовательских запросов	✓	
	Управление записями ресурсов пользовательских запросов	✓	

Разрешения для работы с пространствами

Разрешения для работы с пространствами сгруппированы в разделе **Пространства** дерева разрешений. Они определяют, может ли пользователь управлять пространствами и сервисами в системе.

Разрешениями для работы с пространствами обладают только стандартные роли **Администратор** и **Инженер**.

Группа разрешений	Разрешение
Пространства и сервисы	• Создание пространства • Удаление пространства • Просмотр пространства • Изменение пространства • Просмотр сервиса • Запустить сервис • Остановить сервис

Разрешения для работы с уязвимостями

Разрешения для работы с уязвимостями сгруппированы в разделе **Уязвимости** дерева разрешений. Они определяют, может ли пользователь работать с функционалом сканирования на уязвимости в системе.

Стандартная роль **Администратор** обладает всеми разрешениями для работы с уязвимостями.

Группа разрешений	Разрешения
Учетные записи	• Создание учетных записей сканирования • Удаление учетных записей сканирования • Изменение учетных записей сканирования • Просмотр учетных записей сканирования
База уязвимостей	• Настройка обновления базы уязвимостей • Обновление базы уязвимостей • Просмотр раздела **База уязвимостей**
Профили	• Создание профилей сканирования • Удаление профилей сканирования • Изменение профилей сканирования • Просмотр профилей сканирования

Группа разрешений	Разрешения
Сканеры	• Создание сканера • Удаление сканера • Удаление сервиса сканера • Настройки сканера • Просмотр раздела **Сканеры**
Задачи	• Создание задачи сканирования • Удаление задач сканирования • Изменение задачи сканирования • Управление задачами сканирования • Просмотр задач сканирования
Проверки соответствия	• Просмотр раздела **Проверки** • Удаление проверки

Разрешения для работы с доменами

Разрешения для работы с доменами сгруппированы в разделе **CMS** дерева разрешений. Они определяют, может ли пользователь управлять доменами в системе.

Разрешениями для работы с доменами обладают только стандартные роли **Администратор** и **Инженер**.

Группа разрешений	Разрешения
Управление доменами	• Просмотр доменов • Регистрация доменов • Обновление доменов • Изменение доменов • Удаление доменов

Разрешения для работы с автоматизацией

Разрешения для работы со средствами автоматизации сгруппированы в разделе **Playbooks** дерева разрешений. К средствам автоматизации относятся:

- запуски;
- резолюции;
- триггеры;
- конфигурации;
- переменные;
- воркеры.

Стандартная роль **Администратор** обладает всеми разрешениями для работы с автоматизацией.

Группа разрешений	Разрешения	Инженер
Автоматизация → Запуски	Просмотр запусков	✓
	Создание запуска	✓
	Остановка запуска	
Автоматизация → Резолюции	Просмотр резолюций	
	Согласование резолюции	
Автоматизация → Триггеры	Просмотр триггеров	✓
	Выключение и включение триггера	
	Создание триггера	✓
	Редактирование триггера	✓
	Удаление триггера	✓
Ресурсы → Конфигурации	Просмотр конфигураций	✓
	Создание конфигурации	
	Редактирование конфигурации	
	Удаление конфигурации	
	Включение и выключение конфигурации	
	Обновление конфигурации	
Ресурсы → Переменные	Просмотр переменных	✓
	Создание переменной	
	Добавление переменной в запусках	✓
	Редактирование переменной	
	Удаление переменной	
	Включение и выключение переменной	
Ресурсы → Воркеры	Просмотр воркеров	✓
	Создание воркера	✓
	Редактирование воркера	
	Удаление воркера	
	Включение и выключение воркера	
	Обновление воркера	

Политики входа и пароля

Данный раздел описывает настройки, применяемые ко всем пользователям системы для создания и изменения учетных данных. Эти настройки управляют как аспектами безопасности пароля, так и

правилами доступа к системе. При одновременном изменении политики несколькими пользователями, применятся изменения, внесенные тем пользователем, который последним сохранит изменения.

Настройки пароля и входа в систему задаются в разделе **Настройки** → **Управление доступом** → **Политики входа и пароля** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Настройка политик входа и пароля

Чтобы настроить политики входа и пароля:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Политики входа и пароля**.
2. Задайте следующие настройки:
   - Требования к паролям
   - Срок действия пароля
   - Политики входа
   - Блокирование неиспользуемой учетной записи
   - Блокирование сеанса пользователя после установленного времени бездействия
3. Чтобы сохранить внесенные изменения, нажмите на кнопку **Сохранить** в нижней части раздела.

   Чтобы отменить все несохраненные изменения, нажмите на кнопку **Сбросить** в нижней части раздела.

Требования к паролям

Если пользователь задаст пароль, не соответствующий установленным требованиям, он увидит соответствующее предупреждение. Также, при изменении требований к паролю, пользователь увидит предупреждение о том, что существующий пароль не соответствует новой парольной политике, и сможет изменить пароль в соответствии с ней.

Для включения политики надежных паролей:

1. Переведите переключатель **Требовать надежные пароли** в активное положение.
2. Задайте минимальную длину пользовательского пароля и установите флажки напротив требуемых условий для создания пароля.
3. Нажмите на кнопку **Сохранить** в нижней части раздела. Система применит изменения и отобразит соответствующее уведомление.

Политика надежных паролей включает следующие требования:

- **Минимальная длина пароля**: задает минимально допустимое количество символов в пароле.
- **Должен содержать**: задает классы символов, которые должен содержать пароль. Можно включить в требования следующие классы:

- строчные латинские буквы (a–z);
- заглавные латинские буквы (A–Z);
- цифры (0–9);
- специальные символы:

```
~"!@#$%^&*_-+=`(){}[]:;'<>,.?/|\
```

Кроме того, можно включить ограничения на содержание пароля:

- запрет на последовательное повторение символов;
- запрет на совпадение пароля с текущим или предыдущим.

Срок действия пароля

Срок действия пароля определяет, на какой срок устанавливается пароль, прежде чем его необходимо будет изменить. Допустимые значения: от 7 до 365 дней. Если переключатель **Срок действия пароля** выключен, пароль действует бессрочно.

Чтобы установить срок действия пароля:

1. Переведите переключатель **Срок действия пароля** в активное положение.
2. Задайте срок действия пароля вручную или с помощью кнопок — и +. Срок действия задается в днях.
3. Нажмите на кнопку **Сохранить** в нижней части раздела. Система применит изменения и отобразит соответствующее уведомление.

   За 7 дней до истечения срока действия система отобразит уведомление о необходимости смены пароля.

Политики входа

Чтобы включить блокировку учетных записей после неправильного ввода пароля:

1. Переведите переключатель **Политики входа** в активное положение.
2. Настройте количество попыток входа для включения блокировки и время, на которое блокируется учетная запись.
   - **Период ввода попыток**: время в минутах, в течение которого пользователь может совершить предельное количество попыток входа из параметра **Количество попыток до блокировки**, прежде чем его учетная запись будет заблокирована. Допустимые значения: от 1 до 60 минут.
   - **Количество попыток до блокировки**: определяет, сколько раз пользователь может ввести неправильный пароль в течение времени из параметра **Период ввода попыток**, прежде чем его учетная запись будет заблокирована. Допустимые значения: от 3 до 10 попыток.

Пример

Если **Период ввода попыток** равен 5 минутам и **Количество попыток до блокировки** равно 3, то учетная запись пользователя будет заблокирована

после 3 неудачных попыток входа, совершенных за 5 минут. При этом если пользователь не достигнет лимита неудачных попыток и подождет более 5 минут после последней попытки, ему снова будут доступны 3 попытки.

- **Блокировать**: позволяет выбрать, будет ли учетная запись заблокирована **бессрочно** или **на время**.
- **Период блокировки**: время в минутах, на которое учетная запись будет заблокирована после превышения допустимого количества попыток входа. Допустимые значения: от 3 до 60 минут. Настройка доступна для редактирования если в поле **Блокировать** выбран вариант **На время**.

Блокирование неиспользуемой учетной записи

Чтобы включить блокирование неиспользуемых учетных записей:

1. Переведите переключатель **Блокирование неиспользуемой учетной записи** в активное положение.
2. Выберите из выпадающего списка единицы времени: часы, дни, месяцы.
3. Задайте период неактивности.

Период неактивности позволяет указать промежуток времени с последнего входа в систему, по истечении которого неактивная учетная запись будет автоматически заблокирована.

Минимальный возможный период неактивности зависит от времени жизни токена обновления (refresh token) и может быть изменен с помощью переменной окружения `EVO_CORE_ACCESS_JWT_REFRESH_TOKEN_EXPIRATION_TIME`.

Блокирование сеанса пользователя после установленного времени бездействия

Период неактивности устанавливает время в минутах, после которого неактивный сеанс пользователя будет автоматически завершен.

Минимальный возможный период неактивности зависит от времени жизни токена доступа (access token) и может быть изменен с помощью переменной окружения `EVO_CORE_ACCESS_JWT_ACCESS_TOKEN_EXPIRATION_TIME`.

Настройка времени жизни для токенов доступа

В системе предусмотрены два вида токенов доступа:

- **Access token** — токен, выдаваемый пользователю при входе в систему. Имеет короткое время жизни.
- **Refresh token** — токен, предназначенный для периодического обновления пары **access/refresh token**. Имеет более продолжительное время жизни, чем **Access token**.

Чтобы изменить время жизни токенов доступа пользователей:

1. Подключитесь к кластеру с помощью Lens.
2. Перейдите в раздел **Config** → **Config Maps**.

3. Выберите пространство имен кластера в выпадающем списке **Select Namespace**. Отобразится таблица конфигураций кластера.
4. Выберите конфигурацию **evo.core-access-env**. В правой части экрана отобразится панель настроек конфигурации.
5. В разделе **Data** панели настроек измените значения следующих переменных окружения:
   - `EVO_CORE_ACCESS_JWT_ACCESS_TOKEN_EXPIRATION_TIME` — время жизни **Access token**.

     Период неактивности для блокирования сеанса пользователя из-за бездействия должен быть не меньше значения `EVO_CORE_ACCESS_JWT_ACCESS_TOKEN_EXPIRATION_TIME`.

   - `EVO_CORE_ACCESS_JWT_REFRESH_TOKEN_EXPIRATION_TIME` — время жизни **Refresh token**.

     Период неактивности для блокирования неиспользуемой учетной записи должен быть не меньше значения `EVO_CORE_ACCESS_JWT_REFRESH_TOKEN_EXPIRATION_TIME`.

     Значение `EVO_CORE_ACCESS_JWT_REFRESH_TOKEN_EXPIRATION_TIME` должно быть строго больше значения `EVO_CORE_ACCESS_JWT_ACCESS_TOKEN_EXPIRATION_TIME`.

   - `SINGLE_SESSION` — логическая переменная:
     - если `SINGLE_SESSION` имеет значение `true`, то для одной учетной записи в системе разрешен только один одновременный сеанс;
     - если `SINGLE_SESSION` имеет значение `false`, то количество одновременных сеансов не ограничено.
6. Перейдите в раздел **Workloads** → **Deployments**.
7. Выберите пространство имен кластера в выпадающем списке **Select Namespace**. Отобразится таблица деплойментов кластера.
8. В поле поиска введите `evo.core.access`.
9. Выберите деплоймент **evo.core.access** в таблице. В правой части экрана откроется панель свойств деплоймента.
10. Чтобы применить настройки, перезагрузите деплоймент, нажав на кнопку **Restart** на панели свойств.

LDAP-соединения

Данный раздел описывает процесс работы с LDAP-соединениями в системе. Работа осуществляется в разделе **Настройки** → **LDAP-соединения** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О LDAP-соединениях

LDAP-соединения используются для интеграции системы со службами каталогов, такими как Microsoft Active Directory, ALD Pro, FreeIPA, OpenLDAP. Использование LDAP-соединений позволяет управлять учетными записями и доступом пользователей к ресурсам системы.

LDAP (Lightweight Directory Access Protocol) позволяет централизованно управлять доступом пользователей и их правами в системе. В системе LDAP-соединения обеспечивают:

- аутентификацию пользователей с использованием внешних служб каталогов;
- синхронизацию данных пользователей, включая логин, ФИО, должность, статус, электронный адрес и номер телефона;
- управление ролями и разрешениями пользователей, импортированных из внешних каталогов.

Поддерживаются следующие типы каталогов:

- **Active Directory** — каталоговая служба Microsoft Active Directory.
- **ALD Pro** — каталоговая служба Astra Linux.
- **FreeIPA** — система управления идентификацией и аутентификацией для Linux.
- **OpenLDAP** — реализация протокола LDAP с открытым исходным кодом.

Интерфейс раздела

Панель инструментов раздела включает следующие компоненты:

- Кнопка **Создать** (+) позволяет добавить новое LDAP-соединение.
- Кнопка **Удалить** (🗑) позволяет удалить выбранное LDAP-соединение.
- Поле **Поиск** предназначено для быстрого поиска LDAP-соединений в отображаемом списке по названию.

В рабочей области отображается таблица имеющихся в системе LDAP-соединений, которые доступны пользователю. Таблица содержит следующие столбцы:

- **Название** — имя LDAP-соединения.
- **Статус** — текущий статус соединения: **Включено** или **Нет соединения**.
- **Домен** — домен, к которому подключено соединение.
- **Тип** — тип каталога (например, Active Directory).
- **Описание** — краткое описание соединения.
- **Дата создания** — дата и время создания соединения.
- **Дата изменения** — дата и время последнего изменения соединения.

При работе с таблицей доступны следующие операции:

- Поиск LDAP-соединений по полям **Название** и **Домен**.
- Сортировка LDAP-соединений по полям **Название**, **Домен**, **Дата создания** и **Дата изменения**.

При выборе конкретного соединения в правой части рабочей области отображается его карточка с детальной информацией.

Работа с LDAP-соединением

Доступные операции над LDAP-соединением:

- Создание LDAP-соединения
- Просмотр LDAP-соединения
- Синхронизация LDAP-соединения
- Изменение LDAP-соединения
- Удаление LDAP-соединения

Создание LDAP-соединения

Вы можете создать новое LDAP-соединение для интеграции со службами каталогов, такими как Microsoft Active Directory, ALD Pro, FreeIPA, OpenLDAP.

Чтобы создать новое LDAP-соединение:

1. Перейдите в раздел **Настройки** → **LDAP-соединения**. Система отобразит сведения об имеющихся соединениях.
2. Нажмите на кнопку **Создать** (＋). Отобразится окно добавления LDAP-соединения.
3. Заполните следующие поля:
   - **Название** — введите название LDAP-соединения.
   - **Тип** — выберите тип каталога из выпадающего списка:
     - **Active Directory** — каталоговая служба Microsoft Active Directory.
     - **ALD Pro** — каталоговая служба Astra Linux.
     - **FreeIPA** — система управления идентификацией и аутентификацией для Linux.
     - **OpenLDAP** — реализация протокола LDAP с открытым исходным кодом.
   - **Домен** — введите домен LDAP-соединения. Например: `mycompany.com`.
   - **Адрес LDAP-сервера** — введите IP-адрес или полное доменное имя (FQDN) LDAP-сервера. Например: `ldap.mycompany.com` или `192.168.1.1`.
   - **Порт** — введите номер порта. По умолчанию используются 389 или 636 (при использовании SSL).
   - **Использовать SSL** — установите этот флажок, если необходимо использовать SSL для защищенного соединения.
   - **Имя пользователя** — введите имя пользователя для подключения к LDAP-серверу.
   - **Пароль** — введите пароль пользователя.
   - **Базовый DN** — введите базовый DN (Distinguished Name) для поиска пользователей и групп. Например: `OU=users,DC=mycompany,DC=com`. См. Основные пользовательские атрибуты Active Directory
   - **Описание** (опционально) — введите описание соединения.

▼ *Пример настроек LDAP-соединения*

Название	Пример настройки
Тип	Active Directory
Домен	example.com
Адрес LDAP сервера	192.168.1.1
Порт	636
Использовать SSL	[✓]
Имя пользователя	admin@example.com
Пароль	password123
Базовый DN	dc=example,dc=com
Описание	Соединение для тестового домена

4. Нажмите на кнопку **Проверить подключение** (🔄). Система проверит настройки подключения к выбранной службе каталогов и отобразит статус подключения: **Подключение успешно** или **Ошибка подключения**.
5. Нажмите на кнопку **Создать**. Система создаст LDAP-соединение и отобразит соответствующее уведомление. Новое соединение отобразится в списке раздела.

Просмотр LDAP-соединения

Чтобы просмотреть LDAP-соединение:

1. Перейдите в раздел **Настройки** → **LDAP-соединения**. Система отобразит сведения об имеющихся соединениях.
2. Нажмите на строку соединения в списке. Система отобразит в правой части экрана карточку этого соединения с подробной информацией о нем.

В верхней части карточки отображается название LDAP-соединения. Справа от названия расположена кнопка ⋮, по которой открывается выпадающее меню с действиями, доступными для выбранного соединения.

Карточка содержит основные сведения о LDAP-соединении:

- **Название** — имя LDAP-соединения.
- **Статус** — текущий статус соединения: **Включен** или **Нет соединения**.
- **Тип** — тип каталога (например, Active Directory).
- **Домен** — домен, к которому подключено соединение.
- **Адрес LDAP-сервера** — адрес сервера LDAP.
- **Порт** — номер порта для подключения к серверу LDAP.
- **SSL** — информация о том, используется ли SSL для соединения.
- **Имя пользователя** — имя пользователя для подключения к серверу LDAP.
- **Базовый DN** — базовый DN для поиска пользователей и групп.

- **Описание** — краткое описание соединения.
- **Дата создания** — дата и время создания соединения.
- **Создал** — пользователь, инициировавший создание соединения.
- **Дата изменения** — дата и время последнего изменения соединения.
- **Изменил** — пользователь, выполнивший последнее изменение соединения.
- **Синхронизировано** — информация о последней синхронизации данных с LDAP-сервером.

В нижней части карточки также расположена кнопка **Синхронизировать** (), позволяющая обновить данные пользователей из LDAP-сервера.

Синхронизация LDAP-соединения

Синхронизация обновляет данные пользователей из LDAP-сервера в системе.

Чтобы синхронизировать данные:

1. Перейдите в раздел **Настройки** → **LDAP-соединения**. Система отобразит сведения об имеющихся соединениях.
2. Нажмите на строку соединения в списке. Система отобразит в правой части экрана карточку этого соединения с подробной информацией о нем.
3. Нажмите на кнопку **Синхронизировать** () в нижней части карточки. Система выполнит синхронизацию данных с сервером LDAP и отобразит уведомление о статусе синхронизации.

Изменение LDAP-соединения

Чтобы изменить параметры LDAP-соединения:

1. Перейдите в раздел **Настройки** → **LDAP-соединения**. Система отобразит сведения об имеющихся соединениях.
2. Нажмите на строку соединения в списке. Система отобразит в правой части экрана карточку этого соединения с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно изменения LDAP-соединения.
4. Внесите требуемые изменения.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке соединения и в списке раздела.

Удаление LDAP-соединения

При удалении LDAP-соединения все учетные записи пользователей, связанные с этим доменом, также будут удалены.

Чтобы удалить LDAP-соединение:

1. Перейдите в раздел **Настройки** → **LDAP-соединения**. Система отобразит сведения об имеющихся соединениях.

2. Нажмите на строку соединения в списке. Система отобразит в правой части экрана карточку этого соединения с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Система отобразит окно подтверждения удаления.
4. Нажмите на кнопку **Удалить**. Система удалит соединение и отобразит соответствующее уведомление. Удаленное соединение будет исключено из списка.

Вы также можете удалить сразу несколько соединений. Для этого:

1. Перейдите в раздел **Настройки** → **LDAP-соединения**. Система отобразит сведения о доступных соединениях.
2. Установите флажки напротив тех соединений, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Система отобразит окно подтверждения удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные соединения и отобразит соответствующее уведомление. Удаленные соединения будет исключены из списка.

Основные пользовательские атрибуты Active Directory

OU (Organizational Unit) — Подразделение

Атрибут	Англоязычное название	Русскоязычное название	Пример
`distinguishedName`	Distinguished Name	Отличительное (уникальное) имя	OU=Компания, DC=domain, DC=com
`name`			Компания

Group — Группа

Атрибут	Англоязычное название	Русскоязычное название	Пример
`distinguishedName`	Distinguished Name	Отличительное (уникальное) имя	CN=Группа, OU=Компания, DC=domain, DC=com
`name`			Группа
`member`	Members	Члены группы (какие пользователи входят в данную группу)	CN=Сергей Петрович Иванов, OU=Компания, DC=domain, DC=com

User — Пользователь

Атрибут	Англоязычное название	Русскоязычное название	Пример
DN	Distinguished Name	Отличительное (уникальное) имя	CN=NW_Admins, OU=EvilCorp, DC=domain, DC=com
DC	Domain Component	Компонент (класс) доменного имени	DC=domain, DC=com
OU	Organizational Unit	Подразделение	EvilCorp
CN	Common Name	Общее имя	NW_Admins
givenName	First name	Имя	Сергей Петрович
sn (SurName)	Last name	Фамилия	Иванов
name	Full name	Полное имя	Сергей Петрович Иванов
displayName	Display Name	Отображаемое имя	Сергей Петрович Иванов
mail	E-mail	Электронная почта	sivanov@domain.com
sAMAccountName	User logon name (pre-Windows 2000)	Имя входа пользователя (до-Windows 2000)	IvanovSP
userPrincipalName	User logon name	Имя входа пользователя	IvanovSP@domain.com
memberOf	Member Of	Членство в группах	CN=Группа, OU=Компания, DC=domain, DC=com
AccountPassword	Password	Пароль	

Политики управления доступом к событиям

Данный раздел описывает процесс настройки политик доступа пользователей к событиям в хранилищах на основе их ролей. Работа осуществляется в разделе **Инструменты** → **Политики управления доступом к событиям** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О политиках управления доступом к событиям

Политика управления доступом к событиям — это правило, на основе которого пользователи, имеющие заданные роли, будут иметь доступ только к определенным событиям в хранилище. Правило формулируется как условие фильтрации на языке RQL, которое будет неявно добавляться в секцию WHERE поисковых запросов пользователей.

Составление условия фильтрации из правил

Политики, распространяющиеся на роли текущего пользователя, добавляются к запросу согласно следующей логике:

- Правила внутри одной политики соединяются оператором `OR`.
- Политики соединяются между собой оператором `AND`.
- Все политики соединяются с запросом оператором `AND`.

Рассмотрим следующий пример:

- Пользователь ввел следующий запрос в разделе **Поиск**:

  ```
  dvendor = 'Linux'
  ```

- Первая политика задает для этого пользователя следующие правила доступа:

  ```
  dproduct = 'R-Vision'
  at = 'Endpoint'
  ```

- Вторая политика задает следующее правило доступа:

  ```
  app = 'HTTPS'
  ```

Запрос, отображаемый в строке поиска, и запрос, который будет в действительности выполнен, соответствуют друг другу следующим образом:

Запрос в строке поиска	Запрос, который будет выполнен
`dvendor = 'Linux'`	`dvendor = 'Linux' AND ((dproduct = 'R-Vision' OR at = 'Endpoint') AND app = 'HTTPS')`

Интерфейс раздела

Панель инструментов раздела включает следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новую политику.
- Кнопка **Удалить** ( ) позволяет удалить выбранные политики.
- Поле **Поиск** предназначено для быстрого поиска политик в отображаемом списке по названию.
- Кнопка **Фильтр** ( ) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка политик.

В рабочей области отображается таблица существующих политик. Таблица содержит следующие столбцы:

- **Название** — название политики, используемое для ее идентификации.
- **Роли** — список ролей, на которые распространяется политика.
- **Описание** — краткое описание политики.
- **Хранилища** — список хранилищ событий, на которые распространяется политика.
- **Дата создания** — дата и время создания политики.
- **Дата изменения** — дата и время последнего изменения политики.

При работе с таблицей политик доступны следующие операции:

- Поиск политик по полю **Название**.
- Фильтрация политик по полям **Роли** и **Хранилища**. Роли и хранилища выбираются из выпадающих списков.
- Сортировка политик по полям **Название**, **Описание**, **Дата создания**, **Дата изменения**.
- Настройка отображения таблицы политик.

При выборе конкретной политики в правой части рабочей области отображается ее карточка с детальной информацией.

Работа с политикой управления доступом к событиям

Доступные операции над политикой:

- Создание политики управления доступом к событиям
- Просмотр политики управления доступом к событиям
- Изменение политики управления доступом к событиям
- Удаление политики управления доступом к событиям

Создание политики управления доступом к событиям

Чтобы создать политику:

1. Перейдите в раздел **Инструменты** → **Политики управления доступом к событиям**. Система отобразит сведения об имеющихся политиках.
2. Нажмите на кнопку **Создать** (┼). Отобразится окно создания политики.
3. Введите название политики.
4. При необходимости укажите описание политики.
5. В поле **Роль** выберите роли, на которые будет распространяться политика.
6. Добавьте правила доступа к хранилищам событий.

   Чтобы добавить правило:

   a. Нажмите на кнопку **Добавить правило**. Отобразится панель настроек нового правила.

   По умолчанию в окне уже отображается панель настройки для первого правила.

b. В поле **Хранилище** выберите хранилища событий, на которые будет распространяться правило.

c. В поле **Правило** введите условие на языке RQL, которое будет неявно добавляться к поисковым запросам пользователей.

Например, чтобы разрешить пользователям просматривать только события, поступившие с устройств на ОС Linux из определенной подсети, вы можете задать следующее правило:

```
dvendor = 'Linux' AND sourceIP IN '203.0.113.0/24'
```

Чтобы удалить заданное правило, нажмите на кнопку в правом верхнем углу его панели настроек.

7. Нажмите на кнопку **Создать**. Система создаст политику и отобразит уведомление об успешном создании. Новая политика появится в списке раздела **Инструменты → Политики управления доступом к событиям**.

Просмотр политики управления доступом к событиям

Чтобы просмотреть политику:

1. Перейдите в раздел **Инструменты → Политики управления доступом к событиям**. Система отобразит сведения об имеющихся политиках.
2. Нажмите на строку политики в списке. Система отобразит в правой части экрана карточку этой политики с подробной информацией о ней.

В верхней части карточки отображается название политики. Справа от названия расположена кнопка ⋮, по которой открывается выпадающее меню с действиями, доступными для выбранной политики.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка отображает следующую информацию о политике:

- **ID** — уникальный идентификационный код политики.
- **Название** — системное имя политики, используемое для ее идентификации.
- **Описание** — краткое описание политики.
- **ID тенанта** — идентификатор тенанта, в котором была создана политика.
- **Роли** — список ролей, на которые распространяется политика.
- **Дата создания** — дата и время создания политики.
- **Создал** — пользователь, инициировавший создание политики.
- **Дата изменения** — дата и время последнего изменения политики.
- **Изменил** — пользователь, выполнивший последнее изменение политики.

- **Правила доступа** — список правил, созданных в рамках политики. Для каждого правила указываются хранилища событий, на которые оно распространяется, и условие фильтрации на языке RQL.

Изменение политики управления доступом к событиям

Чтобы изменить политику:

1. Перейдите в раздел **Инструменты** → **Политики управления доступом к событиям**. Система отобразит сведения об имеющихся политиках.
2. Нажмите на строку политики в списке. Система отобразит в правой части экрана карточку этой политики с подробной информацией о ней.
3. Выберите опцию **Изменить** в выпадающем меню кнопки действий (⋮) в верхней части карточки. Отобразится окно настроек политики.
4. Внесите изменения в требуемые поля политики.
5. Нажмите на кнопку **Сохранить**. Измененные данные политики будут сохранены, система отобразит соответствующее уведомление.

Удаление политики управления доступом к событиям

Чтобы удалить политику:

1. Перейдите в раздел **Инструменты** → **Политики управления доступом к событиям**. Система отобразит сведения об имеющихся политиках.
2. Нажмите на строку политики в списке. Система отобразит в правой части экрана карточку этой политики с подробной информацией о ней.
3. Выберите опцию **Удалить** в выпадающем меню кнопки действий (⋮) в верхней части карточки. Отобразится окно подтверждения удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранную политику и отобразит соответствующее уведомление. Удаленная политика будет исключена из списка раздела.

Вы также можете удалить сразу несколько политик. Для этого:

1. Перейдите в раздел **Инструменты** → **Политики управления доступом к событиям**. Система отобразит сведения об имеющихся политиках.
2. В левом столбце таблицы установите флажки напротив тех политик, которые вы хотите удалить.
3. Нажмите на кнопку на панели инструментов. Отобразится окно подтверждения удаления политик.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные политики и отобразит соответствующее уведомление. Удаленные политики будут исключены из списка раздела.

Настройка агентов

Агент обеспечивает мониторинг узла, на котором он установлен, и передает данные в режиме реального времени менеджеру через зашифрованный канал.

В этом разделе приведены сведения о настройке агентов.

Содержание раздела:

- Настройки менеджера агентов
- Сбор инвентарной информации
- Сбор событий на агентах

Настройки менеджера агентов

Данный раздел описывает настройки, применяемые ко всем агентам в системе. Эти настройки позволяют управлять параметрами работы агентов.

Настройки агентов задаются в разделе **Настройки** → **Агенты** веб-интерфейса системы.

Раздел доступен администраторам и учетным записям со статусом **Суперадминистратор**.

Настройка параметров работы агентов

В системе доступна настройка следующих параметров работы для агентов:

- **Максимальное время отсутствия активности от агента**: время в секундах, по истечении которого неактивный агент отключается и переходит в статус **Офлайн**. Значение по умолчанию — 30 секунд.
- **Максимальное время жизни команды/сообщения для агента**: количество дней, по истечении которого отправленная команда, не получившая ответа от агента, удаляется из очереди. Значение по умолчанию — 7 дней.

Сбор инвентарной информации

Система позволяет собирать следующие инвентарные сведения об агенте:

- Аппаратная конфигурация: материнская плата, процессоры, память, диски, дисплеи, видеокарты.
- Процессы: состояние и идентификаторы процессов, использование ресурсов.
- Пользователи и группы.
- Сетевых интерфейсы и открытые порты.
- Общие папки и точки монтирования.
- Установленные пакеты и приложения.
- Docker-контейнеры.

Сбор инвентарной информации осуществляется следующим образом:

- При запуске или перезапуске агента.
- При запросе от менеджера агентов.
- По расписанию, заданному в секции `inventory` файла конфигурации rpoint.conf.

Сбор событий на агентах

Агент позволяет детектировать следующие события:

- входы и выходы пользователей;
- создание процессов;
- добавление или изменение сервисов;
- управление пользователями;
- действия с объектами.

Сбор событий настраивается в конфигурации группы агентов.

Структура событий на агентах

Вид событий, собираемых на агентах, зависит от заданных настроек конфигурации группы агентов. Тем не менее события имеют единую верхнеуровневую структуру, представленную в таблице ниже.

Поле	Тип данных	Описание
`agent`	Object	Основная информация об агенте, на хосте которого произошло событие. Подробности приведены в описании секции `agent`.
`data`	Object	Основная информация о событии. Структура и набор полей в данном поле зависят от используемой операционной системы на агенте и от типа события.
`full_log`	String	JSON-строка, содержащая сведения о событии из логов агента. Поле может отсутствовать в зависимости от типа события.
`id`	String	Уникальный идентификатор события в формате UUID.
`location`	String	Полный путь до файла логов, в котором хранится информация о событии.
`timestamp`	String	Временная метка возникновения события.

Поля секции `agent`

В секции `agent` хранится основная информация об агенте, на хосте которого произошло событие.

Поле	Тип данных	Описание
`hostname`	String	Имя хоста агента.
`id`	String	Уникальный идентификатор агента в формате UUID.
`ip`	String	IP-адрес агента.
`name`	String	Название агента.

Поле	Тип данных	Описание
os_name	String	Название операционной системы на компьютере агента.
os_version	String	Версия операционной системы на компьютере агента.

Пример события


```
{
  "agent": {
    "hostname": "rpoint-agent-debian.rvision.local",
    "id": "0ccaa170-25a7-5a0e-a748-f52d118833b1",
    "ip": "198.51.100.0",
    "name": "rpoint-agent-debian.rvision.local",
    "os_name": "Debian GNU/Linux",
    "os_version": "12"
  },
  "data": {
    "audit": {
      "_ev": 204,
      "_os": 100,
      "exe": "/usr/bin/bash",
      "id": 5,
      "is_container": false,
      "login_type": "account_switch",
      "prevuid": 1000,
      "prevuname": "user",
      "session": 1833,
      "success": "yes",
      "terminal": "pipe:[33703]",
      "time": 1750077602349,
      "type": "LOGIN",
      "uid": 0,
      "uname": "root"
    }
  },
  "full_log": "type=LOGIN msg=audit(1750077602.349:5): uid=0 type=account_switch
terminal="pipe:[33703]" exe="/usr/bin/bash" success=yes session=1833",
  "id": "b24d3967-0c4b-48f5-9562-3c5c839ded47",
  "location": "logs/login-logoff.json",
  "timestamp": "2025-06-16T15:40:02"
}
```


Аудит

В данном разделе представлена информация о процедурах аудита, используемых для мониторинга системных событий и проверки состояния источников событий в системе.

Содержание раздела:

- Аудит источников событий
  - Политики аудита источников событий
  - Настройка конвейера для аудита коллектора
- Аудит системных событий
  - Модель события аудита
  - Отслеживание действий суперадминистраторов
  - Отслеживание действий над объектами

Аудит источников событий

В данном разделе представлена информация о способах мониторинга и проверки состояния источников событий системы.

Содержание раздела:

- Политики аудита источников событий
- Настройка конвейера для аудита коллектора

Политики аудита источников событий

Данный раздел описывает процесс работы с источниками событий в системе. Работа осуществляется в разделе **Инструменты** → **Аудит источников** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Об аудите источников

В системе можно контролировать состояние источников, из которых поступают события в коллекторы.

Состояние источников оценивается на основе данных о частоте и количестве поступающих событий. Это позволяет обнаружить отклонения, такие как слабые потоки событий или полное прекращение поступления событий. Для отслеживания таких ситуаций используются настраиваемые политики аудита источников. В политике можно задать пороговое значение количества поступающих событий и период времени, за который они должны учитываться.

Запуск политики аудита источников происходит по расписанию, которое определяется параметром **Интервал**. В заданное время система запускает проверку количества событий, поступивших от источника за интервал времени. Если количество событий меньше или равно значению, указанному в параметре **Порог**, происходит срабатывание политики аудита источников. Одну политику аудита источников можно применить к нескольким источникам событий. Однако одному источнику событий может быть назначена только одна политика. Если к источнику не применена политика аудита источников, то по умолчанию установленный интервал проверки составляет 30 минут, а порог срабатывания — 1 событие.

Политики аудита источников событий отображаются в таблице в разделе **Инструменты** → **Аудит**

источников на вкладке **Политики аудита источников**. В этом разделе можно добавлять новые политики и изменять существующие.

Списки источников отображаются в веб-интерфейсе программы в разделе **Инструменты** → **Аудит источников** на вкладке **Список точек входа**. В этом разделе можно изменять и удалять источники событий.

Интерфейс раздела

Панель инструментов раздела включает следующие компоненты:

- Кнопка **Создать** (+) позволяет добавить новую политику аудита источников. Кнопка отображается только на вкладке **Политики аудита источников**.
- Поле **Поиск** предназначено для быстрого поиска источников событий и политик аудита источников по их названию и ID.
- Кнопка **Фильтр** (▽) позволяет применять фильтры для отображения источников событий и политик аудита источников по различным критериям.

Раздел содержит две вкладки:

- **Список точек входа** — отображает список всех подключенных источников событий, их статус, идентификатор, тип, адрес и порт подключения, а также информацию о связанных с ними конвейерах, коллекторах и политиках аудита источников.
- **Политики аудита источников** — предоставляет возможности управления политиками аудита источников, которые применяются к источникам событий, включая создание, редактирование и удаление политик, а также отображение их текущего статуса, пороговых значений, интервалов проверки, даты создания и последнего изменения.

В рабочей области вкладки **Список точек входа** отображается таблица имеющихся в системе источников событий, которые доступны пользователю. Таблица содержит следующие столбцы:

- **Статус** — текущий статус источника событий:
  - **Включен** — коллектор и конвейер, содержащие точку входа, включены и на точку поступают события.
  - **Выключен** — коллектор или конвейер, содержащий точку входа, отключен.
  - **Нарушение политики** — коллектор и конвейер, содержащие точку входа, включены, но на точку входа поступает событий меньше, чем определено ее политикой аудита.

Если политика аудита не задана или отключена, то статус **Нарушение политики** определяется согласно политике по умолчанию со следующими параметрами:

- Порог: 1.
- Интервал: 30 мин.

Таким образом, система будет проверять поступление на точку входа хотя бы одного события за 30 минут.

- **ID элемента** — уникальный идентификатор источника событий.
- **Название** — имя источника событий.
- **Тип точки входа** — тип источника событий (например, Endpoint).
- **Адрес подключения** — IP-адрес или доменное имя источника событий.
- **Порт подключения** — номер порта для подключения к источнику событий.
- **Конвейер** — название и идентификатор конвейера событий.
- **Коллектор** — название и идентификатор коллектора.
- **Политика аудита источников** — название политики аудита источников, назначенной источнику событий.

При работе с таблицей на вкладке **Список точек входа** доступны следующие операции:

- Поиск источников событий по полям **Название** и **ID элемента**.
- Фильтрация источников событий по полям **Тип точки входа**, **Статус**, **Коллектор**, **Конвейер** и **Политика аудита источников**. Значение каждого поля выбирается из выпадающего списка.
- Сортировка источников событий по полям **Название**, **Адрес подключения**, и **Порт подключения**.
- Настройка отображения таблицы.

В рабочей области вкладки **Политики аудита источников** отображается таблица существующих политик аудита источников. Таблица содержит следующие столбцы:

- **Название** — имя политики.
- **Порог** — количество событий, полученных от источника за заданный интервал времени, при достижении которого происходит срабатывание политики.
- **Интервал** — период, за который в политике должны учитываться события, полученные от источника.
- **Дата создания** — дата и время создания политики.
- **Дата изменения** — дата и время последнего изменения политики.
- **Статус** — текущий статус политики (включен/выключен).

При работе с таблицей на вкладке **Политики аудита источников** доступны следующие операции:

- Поиск политик аудита источников по полям **Название** и **ID элемента**.
- Фильтрация политик аудита источников по полю **Статус**. Статус выбирается из выпадающего списка.
- Сортировка политик аудита источников по полям **Название**, **Дата создания**, **Дата изменения**, и **Порог**.
- Настройка отображения таблицы.

При выборе отдельного источника событий или политики аудита в правой части рабочей области отображается карточка с детальной информацией.

Источники событий

Доступные операции на вкладке **Источники событий**:

- Просмотр карточки источника событий
- Переход к конвейеру источника событий
- Переход к коллектору источника событий
- Переход к секрету источника событий
- Выбор политики аудита источников

Просмотр карточки источника событий

Чтобы просмотреть карточку источника событий:

1. Перейдите на вкладку **Список точек входа** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных источниках событий.
2. Нажмите на строку источника в списке. Система отобразит в правой части экрана карточку этого источника с подробной информацией о нем.

В верхней части карточки отображается название источника событий.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка источника событий включает в себя следующие поля:

- **ID** — уникальный идентификатор источника событий.
- **Название** — название источника событий.
- **ID тенанта** — идентификатор тенанта, к которому относится источник.
- **Тип точки входа** — тип источника событий (например, Endpoint).
- **Статус** — текущий статус источника событий:
  - **Включен** — коллектор и конвейер, содержащие точку входа, включены и на точку поступают события.
  - **Выключен** — коллектор или конвейер, содержащий точку входа, отключен.
  - **Нарушение политики** — коллектор и конвейер, содержащие точку входа, включены, но на точку входа поступает событий меньше, чем определено ее политикой аудита.

Если политика аудита не задана или отключена, то статус **Нарушение политики** определяется согласно политике по умолчанию со следующими параметрами:

- Порог: 1.
- Интервал: 30 мин.

Таким образом, система будет проверять поступление на точку входа хотя бы одного события за 30 минут.

- **Адрес подключения** — IP-адрес или доменное имя источника событий.

  Для источников AMQP и Database в поле указывается название секрета, использующегося для задания адреса подключения.

- **Порт точки входа** — номер порта для подключения к источнику событий.
- **Конвейер** — название конвейера обработки событий, к которому относится источник.
- **Коллектор** — название коллектора, к которому относится источник.
- **Политика аудита источников** — название политики аудита источников, назначенной источнику событий.

Переход к конвейеру источника событий

Чтобы перейти к конфигурации конвейера, в котором настроен источник событий:

1. Перейдите на вкладку **Список точек входа** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных источниках событий.
2. Нажмите на строку источника в списке. Система отобразит в правой части экрана карточку этого источника с подробной информацией о нем.
3. Нажмите на значение поля **Конвейер** в карточке источника. Система откроет страницу с конфигурацией конвейера, в котором находится выбранный источник событий.

Переход к коллектору источника событий

Чтобы перейти к коллектору, к которому относится источник событий:

1. Перейдите на вкладку **Список точек входа** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных источниках событий.
2. Нажмите на строку источника в списке. Система отобразит в правой части экрана карточку этого источника с подробной информацией о нем.
3. Нажмите на значение поля **Коллектор** в карточке источника. Система откроет страницу раздела **Коллекторы** и отобразит карточку коллектора, в котором содержится конвейер с выбранным источником событий.

Переход к секрету источника событий

Для источников AMQP и Database адрес подключения задается с помощью секрета типа **Строка подключения**.

Чтобы просмотреть информацию о секрете, использующемся в источнике событий:

1. Перейдите на вкладку **Список точек входа** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных источниках событий.
2. Нажмите на строку источника типа AMQP или Database в списке. Система отобразит в правой части экрана карточку этого источника с подробной информацией о нем.
3. Нажмите на значение поля **Адрес подключения** в карточке источника. Система откроет страницу раздела **Секреты** и отобразит карточку секрета, который используется в выбранном источнике событий.

Выбор политики аудита источников

Чтобы выбрать политику аудита для источника событий:

1. Перейдите на вкладку **Список точек входа** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных источниках событий.
2. Нажмите на строку источника в списке. Система отобразит в правой части экрана карточку этого источника с подробной информацией о нем.
3. Нажмите на кнопку в строке поля **Политика аудита источников** в карточке источника и выберите политику из выпадающего списка. На выбор доступны только включенные политики аудита источников.
4. Нажмите на кнопку **Сохранить**. Выбранная политика применится к источнику событий.

Вы также можете выбирать политику аудита для источника событий:

- при добавлении или изменении источника;
- при добавлении или изменении политики аудита источников.

Политики аудита источников

Доступные операции на вкладке **Политики аудита источников**:

- Добавление политики аудита источников
- Включение политики аудита источников
- Отключение политики аудита источников
- Просмотр политики аудита источников
- Изменение политики аудита источников
- Удаление политики аудита источников

Добавление политики аудита источников

Чтобы добавить политику аудита источников:

1. Перейдите на вкладку **Политики аудита источников** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных политиках аудита источников.
2. Нажмите на кнопку **Создать** (+). Отобразится форма создания политики аудита источников.
3. Заполните необходимые поля:
   - **Название** — введите название политики аудита источников.
   - **Описание** (опционально) — введите описание политики аудита источников.
   - **Порог** — задайте количество событий, при достижении которого будет происходить срабатывание политики аудита источников.
   - **Интервал** — задайте период времени в минутах, часах или днях, за который должны учитываться события из источника для срабатывания политики аудита источников.

Точка входа может быть привязана только к одной политике аудита

источников, но к одной политике может быть привязано несколько точек входа. При привязке новой политики к точке входа, она заменит ранее привязанную к ней политику.

- **Точки входа** (опционально) — выберите элементы, к которым необходимо применить политику аудита источников, из дерева точек входа. Для выбора доступны точки входа всех установленных конвейеров во всех включенных коллекторах.

  Чтобы выбрать точку входа, установите флажок слева от ее названия в списке.

  Точки входа в дереве сгруппированы по конвейерам и коллекторам. Чтобы просмотреть список конвейеров нужного коллектора или список точек входа нужного конвейера, нажмите на кнопку ⌄ справа от названия соответствующего родительского элемента в дереве.

  Чтобы скрыть отображенный список дочерних элементов, нажмите на кнопку ⌃ справа от названия родительского элемента в дереве.

4. Нажмите на кнопку **Создать**. Система создаст новую политику аудита источников и отобразит ее в списке политик.

Включение политики аудита источников

Включение или отключение политики аудита источников влияет на ее работоспособность и доступность в списках выбора политик при создании точки входа. Когда политика аудита источников включена, она активно применяется к привязанным точкам входа и доступна для выбора. При отключении политики аудита источников, она перестает применяться к привязанным точкам входа и не отображается в списках выбора политик точек входа.

Чтобы включить политику аудита источников:

1. Перейдите на вкладку **Политики аудита источников** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных политиках аудита источников.
2. Нажмите на строку политики в списке. Система отобразит в правой части экрана карточку этой политики с подробной информацией о ней.
3. Переведите переключатель состояния элемента в верхней части карточки в активное положение. Отобразится окно подтверждения включения политики аудита источников.
4. Нажмите на кнопку **Включить**. Система обновит информацию о статусе политики аудита источников, и она станет активной.

Отключение политики аудита источников

Чтобы отключить политику аудита источников:

1. Перейдите на вкладку **Политики аудита источников** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных политиках аудита источников.
2. Нажмите на строку политики в списке. Система отобразит в правой части экрана карточку этой политики с подробной информацией о ней.
3. Переведите переключатель состояния элемента в верхней части карточки в неактивное положение. Отобразится окно подтверждения отключения политики аудита источников.

4. Нажмите на кнопку **Отключить**. Система обновит информацию о статусе политики аудита источников, и она станет неактивной.

Просмотр политики аудита источников

Чтобы просмотреть карточку политики аудита источников:

1. Перейдите на вкладку **Политики аудита источников** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных политиках аудита источников.
2. Нажмите на строку политики в списке. Система отобразит в правой части экрана карточку этой политики с подробной информацией о ней.

В верхней части карточки отображается название политики аудита источников.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка политики аудита источников включает в себя следующие поля:

- **ID** — уникальный идентификатор политики аудита источников.
- **Название** — имя политики аудита источников.
- **Описание** — описание политики аудита источников.
- **ID тенанта** — идентификатор тенанта.
- **Статус** — текущее состояние политики аудита источников (Включена/Выключена).
- **Тип политики** — тип точки входа.
- **Порог** — количество событий, полученных от источника за заданный интервал времени, при достижении которого происходит срабатывание политики.
- **Интервал** — период, за который в политике аудита источников должны учитываться события, полученные от источника.
- **Дата создания** — дата и время создания политики аудита источников.
- **Дата изменения** — дата и время последнего изменения политики аудита источников.
- **Создал** — имя пользователя, создавшего политику аудита источников.
- **Изменил** — имя пользователя, изменившего политику аудита источников.
- **Точки входа** — список точек входа, к которым применяется политика аудита источников. Список представлен в виде таблицы со столбцом **Название**.

  Список точек входа можно сортировать по названию.

В верхнем правом углу карточки расположена кнопка действий (⋮). При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранной политики аудита источников: изменение и удаление.

Также в карточке находится переключатель статуса политики аудита источников.

Изменение политики аудита источников

Чтобы изменить параметры существующей политики аудита источников:

1. Перейдите на вкладку **Политики аудита источников** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных политиках аудита источников.
2. Нажмите на строку политики в списке. Система отобразит в правой части экрана карточку этой политики с подробной информацией о ней.
3. Нажмите на кнопку действий в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно редактирования политики аудита источников.
4. Внесите необходимые изменения в поля.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в списке политик.

Изменить название и описание политики можно непосредственно из ее карточки:

1. Перейдите на вкладку **Политики аудита источников** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных политиках аудита источников.
2. Нажмите на строку политики в списке. Система отобразит в правой части экрана карточку этой политики с подробной информацией о ней.
3. Нажмите на кнопку в строке названия или описания политики в ее карточке. Система отобразит поле ввода значения.
4. Введите новое значение поля.
5. Нажмите **Сохранить**. Измененные данные политики аудита источников будут сохранены.

Удаление политики аудита источников

Невозможно удалить предустановленные политики аудита источников, а также политики, назначенные источникам событий.

Чтобы удалить существующую политику аудита источников:

1. Перейдите на вкладку **Политики аудита источников** в разделе **Инструменты** → **Аудит источников**. Система отобразит сведения о доступных политиках аудита источников.
2. Установите флажки напротив тех политик, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку ( ) на панели инструментов. Отобразится окно подтверждения удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные политики аудита источников и отобразит соответствующее уведомление. Удаленные политики будут исключены из списка.

Удалить политику аудита источников можно также из ее карточки. Для этого:

1. Перейдите на вкладку **Политики аудита источников** в разделе **Инструменты** → **Аудит источников**.
2. Нажмите на строку политики в списке. Система отобразит в правой части экрана карточку этой политики с подробной информацией о ней.
3. Нажмите на кнопку действий в верхнем правом углу карточки и выберите опцию **Удалить**.

Отобразится окно подтверждения удаления политики.

4. Нажмите на кнопку **Удалить**. Система удалит политику и отобразит соответствующее уведомление. Удаленная политика будет исключена из списка.

Настройка параметров политики аудита источников

Порог и интервал определяют частоту срабатывания политики аудита источников и должны настраиваться с учетом ожидаемых пиковых значений поступающих от источника событий.

Если к источнику не применена политика аудита источников, то по умолчанию установленный интервал проверки составляет 30 минут, а порог срабатывания — 1 событие.

Параметр Порог

Порог — количество событий, полученных от источника за заданный интервал времени, при достижении которого происходит срабатывание политики. Если количество событий с точки входа за интервал времени меньше или равно значению порога, происходит срабатывание политики.

Параметр позволяет настроить чувствительность системы к количеству событий, поступающих от источника. Например, при установке порога на 100 событий за час, политика аудита источников будет срабатывать, если за один час от источника не поступит 100 или более событий.

Параметр Интервал

Интервал задает временной промежуток, в течение которого события из источника учитываются для выполнения политики аудита источников. Если количество событий с точки входа за интервал времени больше или равно значению порога, происходит срабатывание политики.

Интервал задает фиксированное время для запуска проверки. Например, если интервал установлен на 1 минуту, система будет запускать политику каждую минуту и проверять количество событий, полученных от источника за эту минуту. Если интервал установлен на 1 час, политика будет запускаться каждый час в начале часа (например, в 15:00, 16:00 и так далее). Если интервал установлен на 1 день, политика будет срабатывать ежедневно в 00:00.

Настройка конвейера для аудита коллектора

Данное руководство описывает процесс настройки конвейера для обнаружения отсутствия ожидаемых событий от точек входа конвейеров выбранного коллектора.

О настройке аудита точек входа коллектора

Чтобы настроить конвейер для мониторинга состояния источников, первоначально требуется создать и настроить конвейер в рамках коллектора, который содержит источники событий с необходимыми метриками для мониторинга. Данные, проходящие через такой конвейер, должны быть отфильтрованы и проанализированы на предмет отсутствия активности от заданных источников.

Для создания конвейера корреляции на отсутствие события, понадобятся следующие элементы:

1. **Сбор метрик**: обеспечивает непрерывный сбор метрик активности от выбранных элементов

коллектора.

2. **Нормализатор**: приводит собранные метрики к единому формату для упрощения последующего анализа.
3. **Коррелятор**: проверяет поступление событий в заданные временные интервалы (настраиваемый параметр `ttl`). Если от элемента события не поступают, генерируется корреляционное событие об их отсутствии. Настройка коррелятора включает определение оптимальных значений для `ttl` и `count` с учетом частоты и природы поступающих данных.
4. **Конечная точка**: отправляет обработанные и коррелированные события в хранилища данных и внешние системы.

Таким образом, настройка конвейера для мониторинга источников событий включает следующие шаги:

1. Добавление конвейера
2. Настройка конфигурации конвейера
   a. Сбор метрик
   b. Нормализация данных
   c. Корреляция данных
   d. Конечная точка
3. Включение коллектора и установка конфигурации конвейера

Добавление конвейера

Конвейер — это упорядоченная последовательность взаимосвязанных элементов, выполняющих отдельные этапы обработки событий в системе.

Добавление и настройка конвейера обработки событий осуществляются в рамках коллектора, который содержит элементы с необходимыми метриками для мониторинга.

Чтобы добавить конвейер:

1. Перейдите на вкладку **Конвейеры** в карточке выбранного коллектора. Система отобразит список конвейеров коллектора.
2. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
3. Введите название конвейера.
4. При необходимости введите описание конвейера.

Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

5. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке конвейеров на вкладке **Конвейеры** в карточке коллектора.

Настройка конфигурации конвейера

Перед настройкой конфигурации конвейера определите, метрики каких источников на конвейерах коллектора требуется отслеживать.

При настройке конвейера необходимо создать его компоненты, сконфигурировать их, упорядочить и объединить в последовательность, соответствующую логике обработки данных.

Чтобы перейти к настройке конфигурации конвейера:

1. Откройте карточку конвейера по стрелке ( ) в его строке и нажмите на кнопку **Конфигурация конвейера** в нижней части карточки.
2. В открытом окне конфигурации конвейера убедитесь, что в выпадающем списке **Версия** в правом верхнем углу окна выбран пункт **Черновик**.

Сбор метрик

Сбор метрик — это элемент конвейера, который собирает метрики по отдельным элементам конвейеров коллектора для их дальнейшей обработки и анализа.

С помощью сбора метрик можно отслеживать поступление событий от источника на конвейер и выявлять ситуации, когда точка входа перестает получать события.

При создании сбора метрик можно задать интервал между сборами метрик в секундах в поле **Период сбора метрик**.

Добавьте на конвейер сбор метрик. При создании выберите элементы для отслеживания и определите интервал между сборами метрик в поле **Период сбора метрик**. Для выбора доступны элементы всех конвейеров в рамках коллектора.

Нормализация данных

На этом этапе метрики, поступающие от различных элементов конвейера, приводятся к единому формату.

Нормализатор преобразовывает события с помощью установленных на нем правил нормализации. Этот элемент конвейера принимает на вход события, проводит их нормализацию и на выходе предоставляет нормализованные события.

Чтобы добавить нормализацию:

1. Добавьте на конвейер нормализатор. При создании элемента, используйте системное правило нормализации "Правило нормализации Мониторинг источников".

   Пример 15. Правило нормализации "Мониторинг источников", схема RObject

   ```
   id: "786d83cb-48e0-4d5a-b050-ebeea34455cd"
   name: "Правило нормализации Мониторинг источников"
   type: normalization_rule
   description: "Правило нормализации для событий сбора метрик"
   version: 0.0.5
   author: R-Vision
   ```

```
tags:
  - metrics

filter: !vrl |
  true

normalizer: !vrl |
  .dproduct = "metrics"
  . |= compact({
    "dvendor": to_string(.raw.namespace) ?? "",
    "name": to_string(.raw.tags.component_id) ?? "",
    "cnt": to_string(.raw.counter.value) ?? "",
    "cat" : to_string(.raw.name) ?? "", })

tests:
  # Наименование теста. Должен быть уникальным для каждого теста в рамках правила
  - name: "testing"
    # Список тестовых событий
    events:
      - raw:
{"counter":{"value":1751.0},"kind":"absolute","name":"component_received_event_byte
s_total","namespace":"vector","tags":{"component_id":"56c8067a-7f1f-4e4c-8095-
f8261ed8227d_WXAm_yVi0VM90-
btxewDl","component_kind":"transform","component_type":"remap","host":"collector-
bf157c1b-107d-4090-b137-65c89fa14933-795dbc6cff-mhckd"},"timestamp":"2024-05-
29T10:26:42.688089484Z"}
    # Проверка для тестовых событий
    assertion: !vrl |
      assert_eq!(.[0].cat, "component_received_event_bytes_total")
      assert_eq!(.[0].cnt, "1751")
      assert_eq!(.[0].dvendor, "vector")
      assert_eq!(.[0].name, "56c8067a-7f1f-4e4c-8095-f8261ed8227d_WXAm_yVi0VM90-
btxewDl")
```

▼ *Интерпретация*

В блоке `normalizer` правила применяется VRL-функция `compact` для создания упрощенной структуры данных. Она включает следующие поля:

- `dvendor`: присваивает данному полю строковое представление значения поля `namespace` из исходного события. Если поле `namespace` отсутствует, присваивается пустая строка.
- `name`: присваивает данному полю строковое представление значения `component_id`, содержащегося в тегах исходного события. Если значение `component_id` отсутствует, возвращается пустая строка.
- `cnt`: задается строковое представление значения счетчика `counter.value` из исходного события. Если значение счетчика отсутствует, возвращается пустая строка.
- `cat`: строковое представление поля name из исходного события. Если поле name не определено, результатом будет пустая строка.

Функция `compact` автоматически удаляет из результата любые ключи,

которым присвоены пустые строки или значения `null`, тем самым создавая более простую для обработки структуру данных.

2. Выход элемента **Сбор метрик** соедините со входом элемента **Нормализатор**.

Корреляция данных

Чтобы события, поступающие на конвейер, коррелировались, необходимо добавить на конвейер коррелятор.

Коррелятор позволяет связывать различные события и создавать комплексные правила на их основе. На входе применяет правила корреляции к потоку событий. На выходе выдает корреляционные события.

Чтобы добавить коррелятор, необходимо установить в нем правило корреляции. Правила корреляции определяют, как коррелятор будет обрабатывать входящие данные и формировать на их основе корреляционные события.

1. Добавьте на конвейер коррелятор. При создании элемента, используйте системное правило корреляции "Правило корреляции Мониторинг источников".

Пример 16. Правило корреляции "Мониторинг источников", схема RObject

```
id: "6cd640a1-c065-40fc-9341-78067ccdefe6"
name: "Правило корреляции Мониторинг источников"
type: correlation_rule
description: "Правило корреляции ориентировано на события с элемента Сбор метрик.
При отсутствии событий с точки входа в течении заданного периода, будет создано
корреляционное событие"
author: R-Vision
version: 0.0.16
known_false_positives:
  - "события приходят блоком данных с определенной периодичностью, которая выше
ttl"
severity: high
tags:
  - metrics

filter: !vrl |
  .dvendor == "vector"

aliases:
  event_1:
    filter: !vrl |
      .cat == "component_received_events_total" && .cnt != "0"

select:
  alias: event_1

group:
  - alias: event_1
```

```
    by:
      - cnt
      - name
    count: 3

ttl: 60

on_correlate: !vrl |
  .dvendor = "vector"
  cs = [ encode_json(%event_1[0].raw), encode_json(%event_1[1].raw),
encode_json(%event_1[2].raw)]
  .cs1 = join(cs, ", ") ?? ""
  .cs1Label = "Базовые события"
  .cnt = to_string(%event_1[0].cnt) ?? "-"
  name = to_string(%event_1[0].name) ?? "-"
  .name = name
  .message = "Перестали поступать события с источника {{name}}"

tests:
# Наименование теста. Должен быть уникальным для каждого теста в рамках правила
  - name: Test event
# Список тестовых событий
    events:
     - {"id":"2f81df63-c2ea-429e-b16d-
cdd1cc1d7a63","sourceIp":"","tenantId":"00000000-0000-0000-0000-
000000000000","art":"2024-06-
04T14:02:18.000+03:00","raw":"{\"counter\":{\"value\":13.0},\"kind\":\"absolute\",\
"name\":\"component_received_events_total\",\"namespace\":\"vector\",\"tags\":{\"co
mponent_id\":\"56c8067a-7f1f-4e4c-8095-f8261ed8227d_WXAm_yVi0VM90-
btxewDl\",\"component_kind\":\"transform\",\"component_type\":\"remap\",\"host\":\"
collector-bf157c1b-107d-4090-b137-65c89fa14933-85c5cb5c6f-
qnh97\"},\"timestamp\":\"2024-06-
04T11:02:18.582264635Z\"}","collectorId":"bf157c1b-107d-4090-b137-
65c89fa14933","timestamp":"2024-06-
04T14:02:18.000+03:00","cnt":13,"type":"нормализованное
событие","dvendor":"vector","dproduct":"metrics","name":"56c8067a-7f1f-4e4c-8095-
f8261ed8227d_WXAm_yVi0VM90-btxewDl","cat":"component_received_events_total"}
     - {"id":"2f81df63-c2ea-429e-b16d-
cdd1cc1d7a63","sourceIp":"","tenantId":"00000000-0000-0000-0000-
000000000000","art":"2024-06-
04T14:02:18.000+03:00","raw":"{\"counter\":{\"value\":13.0},\"kind\":\"absolute\",\
"name\":\"component_received_events_total\",\"namespace\":\"vector\",\"tags\":{\"co
mponent_id\":\"56c8067a-7f1f-4e4c-8095-f8261ed8227d_WXAm_yVi0VM90-
btxewDl\",\"component_kind\":\"transform\",\"component_type\":\"remap\",\"host\":\"
collector-bf157c1b-107d-4090-b137-65c89fa14933-85c5cb5c6f-
qnh97\"},\"timestamp\":\"2024-06-
04T11:02:18.582264635Z\"}","collectorId":"bf157c1b-107d-4090-b137-
65c89fa14933","timestamp":"2024-06-
04T14:02:18.000+03:00","cnt":13,"type":"нормализованное
событие","dvendor":"vector","dproduct":"metrics","name":"56c8067a-7f1f-4e4c-8095-
f8261ed8227d_WXAm_yVi0VM90-btxewDl","cat":"component_received_events_total"}
```

```
    - {"id":"22daee10-9148-4284-be42-
3c0e9c2b9f33","sourceIp":"","tenantId":"00000000-0000-0000-0000-
000000000000","art":"2024-06-
04T14:01:48.000+03:00","raw":"{\"counter\":{\"value\":13.0},\"kind\":\"absolute\",\
"name\":\"component_received_events_total\",\"namespace\":\"vector\",\"tags\":{\"co
mponent_id\":\"56c8067a-7f1f-4e4c-8095-f8261ed8227d_WXAm_yVi0VM90-
btxewDl\",\"component_kind\":\"transform\",\"component_type\":\"remap\",\"host\":\"
collector-bf157c1b-107d-4090-b137-65c89fa14933-85c5cb5c6f-
qnh97\"},\"timestamp\":\"2024-06-
04T11:01:48.582435089Z\"}","collectorId":"bf157c1b-107d-4090-b137-
65c89fa14933","timestamp":"2024-06-
04T14:01:48.000+03:00","cnt":13,"type":"нормализованное
событие","dvendor":"vector","dproduct":"metrics","name":"56c8067a-7f1f-4e4c-8095-
f8261ed8227d_WXAm_yVi0VM90-btxewDl","cat":"component_received_events_total"}

    assertion: !vrl |
      assert_eq!(.[0].message, "Перестали поступать события с источника 56c8067a-
7f1f-4e4c-8095-f8261ed8227d_WXAm_yVi0VM90-btxewDl")
      assert_eq!(.[0].cnt, "13")
      assert_eq!(.[0].dvendor, "vector")
      assert_eq!(.[0].severity, "high")
```

▼ *Интерпретация*

Правило корреляции "Мониторинг источников" предназначено для отслеживания метрик конвейеров коллектора. Если события с точки входа не поступают в течение 1 минуты, коррелятор генерирует корреляционное событие.

Логика фильтрации (filter)

- Основной фильтр — `.dvendor == "vector"`, выбирающий события, у которых значение `dvendor` равно "vector".

Логика фильтрации (aliases)

- Псевдоним `event_1`: фильтр для событий с метрикой, где `cat` равно "component_received_events_total" и `cnt` не равно "0".

Соединение потоков событий (select и group)

- `select`: выбор псевдонима для дальнейшей обработки. В данном случае используется `event_1`.
- `group`: группировка событий по полям `cnt` и `name`. Группировка применяется к псевдониму `event_1` и формируется при накоплении 3 событий.

Срок хранения промежуточных данных корреляции (ttl)

- `ttl` на уровне правила задает общее максимальное время жизни промежуточных корреляционных данных — 60 секунд. Это время определяет, как долго промежуточные результаты будут храниться в памяти до их автоматического удаления.

Пост-корреляционная логика (on_correlate)

При срабатывании корреляции создается сообщение об отсутствии событий с указанного источника.

- Формирование сообщения:
  - `message`: формируется сообщение "Перестали поступать события с источника " + значение `name`, которое указывает на идентификатор элемента, из которого ожидались данные.
  - `dvendor`: поле устанавливается в значение "vector", что соответствует источнику данных.
  - `cnt`: выводится значение счетчика последнего события, что помогает определить объем данных до прекращения поступления.
  - `name`: содержит идентификаторы конвейера и элемента, которые породили событие.
  - `component_id`: отображает идентификатор элемента, предоставляющий конкретный источник данных.

2. Выход элемента **Нормализатор** соедините со входом элемента **Коррелятор**.

Конечная точка

Конечная точка, в зависимости от настроек, пересылает событие в хранилище событий или внешнюю систему. Имеет один вход и не имеет выходов.

1. Добавьте на конвейер конечную точку, в которой будут сохраняться события.
2. Выход элемента **Коррелятор** соедините со входом элемента **Конечная точка**.

Включение коллектора и установка конфигурации конвейера

Перед включением конвейера убедитесь, что связанный с ним коллектор уже включен.

Чтобы метрики начали поступать на конвейер и обрабатываться:

1. Перейдите в карточку коллектора и включите его.
2. Откройте конфигурацию конвейера и установите текущую конфигурацию.

Рекомендации по настройке

Параметры `ttl`, `count` и период сбора метрик влияют на чувствительность и время реакции системы. Важно настроить их в соответствии с конкретными требованиями, чтобы обеспечить баланс между чувствительностью системы и устойчивостью к ложным срабатываниям.

Параметр `ttl`

`ttl` (time-to-live) определяет максимальное время, в течение которого промежуточные результаты корреляционных операций сохраняются в памяти. Значение `ttl` должно соответствовать максимальному времени между ожидаемыми событиями, с учетом допустимой погрешности для обработки возможных задержек.

Чем выше значение `ttl`, тем более длительный период времени охватывает корреляционное окно, что может привести к более позднему обнаружению проблем. Напротив, более низкое значение `ttl`

может привести к частым ложным срабатываниям, особенно в периоды низкой активности или вариативности данных.

Конфигурация `ttl` должна соответствовать максимальному времени между ожидаемыми событиями с учетом некоторой погрешности для обработки возможных задержек.

Формула для настройки `ttl`:

```
ttl = максимальное время между событиями + допустимая погрешность при реагировании
```

Параметр `count`

`count` определяет количество событий, необходимых для формирования группы в правилах корреляции. Значение `count` должно быть выбрано на основе анализа частоты и объема поступающих данных.

Чем выше значение `count`, тем больше событий требуется для срабатывания корреляции, что может уменьшить количество ложных срабатываний, но при этом привести к пропуску важных корреляций. Напротив, более низкое значение `count` увеличивает чувствительность системы, но также может привести к частым ложным срабатываниям.

Период сбора метрик

При создании элемента сбора метрик можно задать интервал между сборами в секундах, что позволяет контролировать частоту мониторинга активности источников. Этот интервал настраивается в поле **Период сбора метрик** при добавлении сбора метрик на конвейер. Его необходимо адаптировать в соответствии с динамикой изменений в отслеживаемой среде.

При выборе интервала важно учитывать не только частоту изменений в данных, но и влияние этого интервала на производительность системы и ресурсы серверов.

Пример настройки параметров `ttl` и `count`

- Для событий, поступающих с периодичностью раз в час, значение `ttl` следует установить на 2 часа для некритичных событий и 1 час 2 минуты для критичных событий.
- Для событий, поступающих каждую секунду, значение `ttl` можно установить на 1 минуту.

Выбор значения параметра `count` зависит от частоты событий:

- Для редких событий значение `count` можно установить на 1, чтобы система реагировала на каждое событие.
- Для частых событий значение `count` можно установить на 5 или выше, чтобы уменьшить количество ложных срабатываний.

Аудит системных событий

О сервисе аудита событий

Сервис аудита предназначен для сбора, анализа и хранения данных о действиях пользователей и событиях в системе. Он обеспечивает контроль за безопасностью, позволяя администраторам

выявлять, хранить и отслеживать все действия пользователей в системе для предотвращения, обнаружения и реагирования на возможные угрозы или нарушения безопасности. С помощью сервиса аудита можно фиксировать такие действия, как входы в систему, создание, удаление и изменение ресурсов системы и конфигурации ее отдельных компонентов.

Сервис аудита работает таким образом, что каждое действие пользователя в системе идентифицируется как отдельное событие аудита. Это событие затем регистрируется и сохраняется в системное хранилище аудита. Основой для этого хранилища служит системная база данных событий аудита, структура которой определяется системной моделью события аудита.

События аудита систематически записываются в базу данных, обеспечивая возможность их постоянного хранения, систематизации и последующего ретроспективного анализа. События аудита доступны для просмотра в разделе **Поиск**. Доступ к этим данным позволяет просматривать историю действий пользователей, анализировать поведенческие паттерны и, при необходимости, реагировать на подозрительные или несанкционированные действия.

События сервиса аудита можно анализировать и обрабатывать на конвейере с помощью точки входа Audit. Например, вы можете настроить конфигурацию конвейера на выполнение следующих операций:

- Отслеживание действий суперадминистраторов
- Отслеживание действий над объектами

Просмотр списка событий аудита

Чтобы просмотреть события сервиса аудита:

1. Перейдите в раздел **Поиск**.
2. Выберите из выпадающего списка в верхней части раздела **Хранилище событий аудита**.
3. Задайте критерии поиска событий:
   - Для поиска событий по ключевым словам или по RQL-запросам введите запрос в поле поиска.

     При установке курсора в поле поиска отображается панель с историей поиска. Система сохраняет здесь 5 последних поисковых запросов.

   - Для поиска событий за период времени выберите из выпадающего списка способ задания периода:
     i. **Последний период** — задайте значение вручную или с помощью кнопок — и +, после чего выберите единицы времени из выпадающего списка. Поиск событий будет выполняться в указанном диапазоне вплоть до нынешнего момента.
     ii. **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск событий.
     iii. **Часто используемые** — выберите в разделе один из предлагаемых периодов.

     В нижней части выпадающего списка параметров для определения периода расположена секция **История поиска**. Система сохраняет здесь

5 последних поисковых запросов.

4. Нажмите на кнопку **Поиск**. Список событий аудита, соответствующий заданным критериям, отобразится в таблице. Также над таблицей отобразится количество найденных совпадений.

При необходимости вы можете остановить процесс поиска, нажав на кнопку **Стоп**. В разделе будут отображены только те события, что были найдены до прерывания поиска.

Для событий аудита доступны все стандартные операции над событиями.

Типы событий аудита

Тип события аудита зависит от действия пользователя, а также от раздела системы, в котором это действие было выполнено.

Система отслеживает следующие события аудита:

- Системные события аудита
- События аудита пространств
- События аудита работы с событиями ИБ
- События аудита менеджера агентов
- События аудита управления сканированием
- События аудита управления доменами
- События аудита управления автоматизацией

Системные события аудита

Группа событий	Тип события
Аутентификация	• Вход пользователя • Выход пользователя • Неуспешная попытка входа • Попытка входа заблокированного пользователя • Пользователь заблокирован по причине превышения попыток входа
Настройка средств аутентификации	• Смена пароля пользователя • Изменение настроек политики требований к паролю • Изменение настроек политики требований к входу

Группа событий	Тип события
Пользователи	• Создание пользователя • Изменение пользователя • Блокировка пользователя • Разблокировка пользователя • Удаление пользователя
Группы пользователей	• Создание группы пользователей • Изменение группы пользователей • Удаление группы пользователей • Синхронизация группы пользователей
Тенанты	• Создание тенанта • Блокировка тенанта • Разблокировка тенанта • Изменение тенанта • Удаление тенанта
Роли	• Создание роли • Изменение роли • Удаление роли
LDAP-соединения	• Создание LDAP-соединения • Изменение LDAP-соединения • Удаление LDAP-соединения • Синхронизация пользователей домена

События аудита пространств

Группа событий	Тип события
Пространства	• Создание пространства • Изменение пространства • Удаление пространства
Сервисы	• Запуск сервиса • Остановка сервиса

События аудита работы с событиями ИБ

Группа событий	Тип события
Дашборды	• Создание дашборда • Изменение дашборда • Удаление дашборда • Импорт дашборда
Виджеты	• Создание виджета • Изменение виджета • Удаление виджета
Активные списки	• Создание активного списка • Изменение активного списка • Удаление активного списка
Записи активных списков	• Создание записи активного списка • Изменение записи активного списка • Удаление записи активного списка • Импорт записей активного списка
Модели событий	• Создание модели события • Изменение модели события • Удаление модели события • Включение модели события • Выключение модели события • Импорт модели события
Базы данных событий	• Создание базы данных • Изменение базы данных • Удаление базы данных
Хранилища событий	• Создание хранилища событий • Изменение хранилища событий • Удаление хранилища событий
Секреты	• Создание секрета • Изменение секрета • Удаление секрета
Шаблоны отчета	• Создание шаблона отчета • Изменение шаблона отчета • Удаление шаблона отчета
Отчеты	• Генерация отчета • Удаление отчета

Группа событий	Тип события
Элементы экспертизы	• Публикация и изменение элемента экспертизы • Создание черновика элемента экспертизы • Удаление элемента экспертизы • Включение элемента экспертизы • Выключение элемента экспертизы
Каталоги	• Создание каталога • Изменение каталога • Перенос элемента экспертизы • Перенос каталога
Коллекторы	• Создание коллектора • Изменение коллектора • Удаление коллектора • Включение коллектора • Выключение коллектора • Импорт коллектора
Шины	• Создание шины • Изменение шины • Удаление шины
Обогащения	• Добавление обогащения • Изменение обогащения • Удаление обогащения
Конвейеры	• Создание конвейера • Изменение конвейера • Удаление конвейера • Создание копии конвейера • Импорт конвейера • Сброс метрик конвейера
Конфигурации конвейеров	• Установка конфигурации • Отключение конфигурации • Изменение конфигурации

Группа событий	Тип события
Политики мониторинга	• Создание политики мониторинга • Изменение политики мониторинга • Удаление политики мониторинга • Включение политики мониторинга • Выключение политики мониторинга

События аудита менеджера агентов

Раздел	Тип события
Агенты → **Агенты**	• Регистрация агента • Удаление агента • Отключение агента • Обновление инвентарной информации об агенте • Добавление агента в группу • Создание группы с выбранным агентом • Удаление агента из группы
Агенты → **Группы агентов**	• Создание группы агентов • Дублирование группы агентов • Изменение группы агентов • Удаление группы агентов • Изменение конфигурации группы агентов
Агенты → **Шлюзы**	• Создание шлюза • Изменение шлюза • Удаление шлюза • Запуск шлюза • Остановка шлюза
Настройки → **Агенты**	• Изменение параметров работы агента

События аудита управления сканированием

Группа событий	Тип события
Профили	• Создание профиля сканирования • Изменение профиля сканирования • Удаление профиля сканирования

Группа событий	Тип события
Учетные записи	• Создание учетной записи сканирования • Изменение учетной записи сканирования • Удаление учетной записи сканирования • Проверка учетной записи сканирования
Сервис обновления	• Запуск обновления • Загрузка обновления • Распаковка обновления • Проверка обновления • Установка обновления • Завершение обновления
Задачи	• Создание задачи сканирования • Изменение задачи сканирования • Удаление задачи сканирования • Приостановка задачи сканирования • Возобновление задачи сканирования • Отмена задачи сканирования • Запуск задачи сканирования • Завершение задачи сканирования
Сканирования	• Создание сканирования • Изменение сканирования в результате изменения связанной задачи

События аудита управления доменами

Группа событий	Тип события
Домены	• Регистрация домена • Изменение информации о домене • Обновление домена • Удаление домена
Сущности	• Создание сущности • Изменение сущности • Удаление сущности
Связи	• Создание связи • Изменение связи • Удаление связи

Группа событий	Тип события
Объекты	• Создание объекта • Изменение объекта • Удаление объекта • Синхронизация объектов • Экспорт объекта
Экземпляры связи	• Создание экземпляра связи • Изменение экземпляра связи • Удаление экземпляра связи

События аудита управления автоматизацией

Группа событий	Тип события
Запуски	• Создание задания на запуск скрипта или плейбука • Запуск выполнения скрипта или плейбука • Завершение выполнения скрипта или плейбука • Остановка выполнения скрипта или плейбука
Триггеры	• Создание триггера • Изменение триггера • Удаление триггера • Включение триггера • Выключение триггера
Воркеры	• Создание воркера • Изменение воркера • Удаление воркера • Включение воркера • Выключение воркера • Остановка воркера • Обновление версии воркера • Возобновление работы воркера
Переменные	• Создание переменной • Изменение переменной • Удаление переменной • Чтение переменной-секрета • Включение переменной • Выключение переменной

Группа событий	Тип события
Конфигурации	• Создание конфигурации • Изменение конфигурации • Обновление конфигурации • Удаление конфигурации • Включение конфигурации • Выключение конфигурации

Модель события аудита

Модель события аудита предназначена для хранения событий сервиса аудита.

Сервис аудита позволяет собирать, анализировать и хранить данные о действиях пользователей и событиях в системе. Он обеспечивает контроль за безопасностью системы, позволяя обнаруживать возможные угрозы или нарушения безопасности и реагировать на них.

С помощью сервиса аудита можно фиксировать такие действия, как входы в систему, создание, удаление и изменение ресурсов системы и конфигурации ее отдельных компонентов. Каждое действие фиксируется в виде события аудита и сохраняется в специальное системное хранилище. События аудита доступны для просмотра в разделе **Поиск**.

Модель события аудита является встроенной системной моделью и недоступна для изменения или удаления. Она отображается в разделе **Ресурсы → Модели событий**.

При необходимости вы можете создать собственную (пользовательскую) модель события на базе системной модели события аудита.

Поля модели события аудита

Система заполняет поля событий аудита автоматически. Часть полей заполняется только для определенных типов событий и может не содержать значений. Остальные поля универсальны и заполняются независимо от типа события.

Таблица 1. Поля модели событий аудита

Ключ поля	Тип данных	Заполняется системой по умолчанию	Описание
`id`	UUID	Да	Уникальный идентификатор события аудита. Генерируется системой автоматически.
`sourceIp`	IPv6	Да	IP-адрес источника, от которого получено базовое событие.
`tenantId`	LCString	Да	Идентификатор тенанта.

Ключ поля	Тип данных	Заполняется системой по умолчанию	Описание
`art`	DateTime	Да	Дата и время получения события системой-источником или агентом.
`raw`	String	Да	Текст базового (сырого) события.
`collectorId`	LCString	Да	Идентификатор коллектора, получившего событие.
`timestamp`	DateTime	Да	Дата и время получения события коллектором.
`type`	Enum	Да	Тип события: 4 — событие аудита.
`rt`	DateTime	Да	Дата и время получения устройством сведений о наблюдаемом событии.
`eventType`	LCString	Да	Тип события аудита.
`moduleKey`	LCString	Да	Раздел системы, в котором совершено действие.
`eventRiskLevel`	Enum	Да	Уровень угрозы события: • 1 — отладочный (`DEBUG`); • 2 — низкий (`LOW`); • 3 — средний (`MEDIUM`); • 4 — высокий (`HIGH`); • 5 — критический (`CRITICAL`); • 6 — фатальный (`FATAL`); • 7 — аварийный (`ACCIDENT`).
`dvc`	IPv6	Нет	IPv6-адрес устройства, с которого получено событие.
`suser`	JSON-строка	Нет	Информация о пользователе, который совершил действие. Примеры сведений: • ID учетной записи. • ФИО учетной записи. • Наличие прав администратора. • Статус учетной записи.

Ключ поля	Тип данных	Заполняется системой по умолчанию	Описание
`data`	JSON-строка	Да	Информация о сущности, над которой совершено действие. Примеры сведений: • ID сущности. • Тип сущности. • Название сущности. • Версия сущности. • Длительность блокировки. • Количество неуспешных попыток ввода. • Максимальное время действия пароля. • Минимальная длина пароля. • Способ аутентификации. • Тип события безопасности.
`meta`	JSON-строка	Да	Информация о событии аудита. Примеры сведений: • Тип события. • Результат выполнения действия (успешно/неуспешно). • Тип действия (создание, изменение, удаление, запуск/включение, остановка/выключение).

Отслеживание действий суперадминистраторов

Сервис аудита в системе позволяет собирать, анализировать и хранить данные о действиях пользователей и событиях в системе. Эти события можно доставлять на конвейер с помощью точки входа Audit и настраивать их обработку в соответствии с конфигурацией конвейера.

Например, можно настроить конфигурацию конвейера так, чтобы события пользователей с ролью **Суперадминистратор** направлялись в отдельное хранилище или внешнюю систему. Это позволит отслеживать действия пользователей с привилегированными правами для выявления потенциальных злоупотреблений или ошибок, которые могут привести к утечке данных.

Данное руководство описывает процесс настройки конвейера для отслеживания действий суперадминистраторов.

Об отслеживании действий суперадминистраторов

Для настройки конвейера с отслеживанием действий суперадминистраторов понадобятся следующие элементы:

1. **Точка входа Аудит**: обеспечивает непрерывный сбор событий сервиса аудита с выбранными уровнями угрозы.
2. **Фильтр**: отсеивает события, не удовлетворяющие условию фильтрации.
3. **Конечная точка**: отправляет обработанные и коррелированные события в хранилища данных и внешние системы.

Таким образом, настройка конвейера для отслеживания действий суперадминистратора включает следующие шаги:

1. Добавление конвейера
2. Настройка конфигурации конвейера
    a. Шаг 1. Точка входа Audit
    b. Шаг 2. Фильтрация данных
    c. Шаг 3. Конечная точка
    d. Шаг 4. Включение коллектора и установка конфигурации конвейера

Добавление конвейера

Конвейер — это упорядоченная последовательность взаимосвязанных элементов, выполняющих отдельные этапы обработки событий в системе.

Чтобы добавить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора, в котором будет выполняться настройка конвейера. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. В нижней части карточки нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
5. Введите название конвейера.
6. При необходимости введите описание конвейера.

Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

7. Нажмите на кнопку **Добавить**. Система добавит конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке конвейеров на вкладке **Конвейеры** в карточке коллектора.

Настройка конфигурации конвейера

При настройке конвейера необходимо создать его компоненты, сконфигурировать их, упорядочить и объединить в последовательность, соответствующую логике обработки данных.

Чтобы перейти к настройке конфигурации конвейера:

1. Откройте карточку конвейера по стрелке (⌄) в его строке и нажмите на кнопку **Конфигурация конвейера** в нижней части карточки.
2. В открытом окне конфигурации конвейера убедитесь, что в выпадающем списке **Версия** в правом верхнем углу окна выбран пункт **Черновик**.

Шаг 1. Точка входа Audit

Точка входа Audit — это элемент конвейера, который собирает события сервиса аудита и доставляет их на конвейер для дальнейшей обработки и анализа. Точка входа имеет один выход и не имеет входов.

С помощью точки входа Audit можно отслеживать действия пользователей в системе для предотвращения, обнаружения и реагирования на возможные угрозы или нарушения безопасности.

Добавьте на конвейер точку входа типа Audit. При создании выберите необходимые уровни угрозы для фильтрации входящего потока событий аудита.

Шаг 2. Фильтрация данных

Чтобы фильтровать входящие события по наличию роли **Суперадминистратор** у пользователя, выполнившего какое-либо действие в системе, необходимо добавить на конвейер фильтр.

Фильтр — это элемент конвейера, который позволяет сократить объем обрабатываемых событий путем их отсеивания на основе условий, заданных на языке VRL. Имеет один вход и один выход.

1. Добавьте на конвейер фильтр. При добавлении элемента задайте следующее условие фильтрации:

   ```
   .suser.is_super_admin == true
   ```

   Данное условие проверяет, является ли пользователь, совершивший действие, суперадминистратором.

2. Выход из элемента **Точка входа** соедините со входом элемента **Фильтр**.

Шаг 3. Конечная точка

Конечная точка, в зависимости от настроек, пересылает событие в хранилище событий или внешнюю систему. Имеет один вход и не имеет выходов.

1. Добавьте на конвейер конечную точку, в которой будут сохраняться события.
2. Выход из элемента **Фильтр** соедините со входом элемента **Конечная точка**.

Шаг 4. Включение коллектора и установка конфигурации конвейера

Перед включением конвейера убедитесь, что связанный с ним коллектор уже включен.

Чтобы события аудита начали поступать на конвейер и обрабатываться:

1. Перейдите в карточку коллектора и включите его.
2. Откройте конфигурацию конвейера и установите текущую конфигурацию.

   События, порожденные действиями суперадминистраторов, будут направляться в заданное хранилище событий или внешнюю систему.

Отслеживание действий над объектами

Сервис аудита в системе позволяет собирать, анализировать и хранить данные о действиях пользователей и событиях в системе. Эти события можно доставлять на конвейер с помощью точки входа Audit и настраивать их обработку в соответствии с конфигурацией конвейера.

Например, можно настроить конфигурацию конвейера так, чтобы в разные хранилища направлялись события аудита разных типов. Это позволит отслеживать определенные действия над различными сущностями в системе для обнаружения несанкционированных действий и поддержания безопасности инфраструктуры организации. Так, например, можно обнаружить изменение или удаление коллектора, критически важного для организации, и своевременно принять необходимые меры.

Данное руководство описывает процесс настройки конвейера для отслеживания действий над объектами системы.

В представленном руководстве термин "объект" используется для обозначения сущностей системы, например, коллекторов, конвейеров, элементов экспертизы, активных списков.

Об отслеживании действий над объектами

Для отслеживания определенных действий над объектами, первоначально требуется создать хранилища событий, в каждом из которых будут храниться события аудита определенного типа. В представленной инструкции будет рассматриваться отслеживание действий с типами "Создание", "Изменение" и "Удаление". Таким образом, для представленного случая понадобятся четыре хранилища: по одному на каждый из трех типов действий и еще одно — для всех остальных.

Для настройки конвейера с отслеживанием действий над объектами понадобятся следующие элементы:

1. **Точка входа Аудит**: обеспечивает непрерывный сбор событий сервиса аудита с выбранными уровнями угрозы.
2. **Маршрутизатор**: направляет поток событий по разным маршрутам в зависимости от условий фильтрации на этих маршрутах.
3. **Конечные точки**: отправляют обработанные события в хранилища данных.

Таким образом, настройка конвейера для отслеживания действий над объектами включает следующие шаги:

1. Добавление конвейера
2. Настройка конфигурации конвейера
    a. Шаг 1. Точка входа Audit
    b. Шаг 2. Маршрутизация данных
    c. Шаг 3. Конечные точки
    d. Шаг 4. Включение коллектора и установка конфигурации конвейера

Добавление конвейера

Конвейер — это упорядоченная последовательность взаимосвязанных элементов, выполняющих отдельные этапы обработки событий в системе.

Чтобы добавить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора, в котором будет выполняться настройка конвейера. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. В нижней части карточки нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
5. Введите название конвейера.
6. При необходимости введите описание конвейера.

Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

7. Нажмите на кнопку **Добавить**. Система добавит конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке конвейеров на вкладке **Конвейеры** в карточке коллектора.

Настройка конфигурации конвейера

При настройке конвейера необходимо создать его компоненты, сконфигурировать их, упорядочить и объединить в последовательность, соответствующую логике обработки данных.

Чтобы перейти к настройке конфигурации конвейера:

1. Откройте карточку конвейера по стрелке (⌄) в его строке и нажмите на кнопку **Конфигурация конвейера** в нижней части карточки.
2. В открытом окне конфигурации конвейера убедитесь, что в выпадающем списке **Версия** в правом верхнем углу окна выбран пункт **Черновик**.

Шаг 1. Точка входа Audit

Точка входа Audit — это элемент конвейера, который собирает события сервиса аудита и доставляет их на конвейер для дальнейшей обработки и анализа. Точка входа имеет один выход и не имеет входов.

С помощью точки входа Audit можно отслеживать действия пользователей в системе для предотвращения, обнаружения и реагирования на возможные угрозы или нарушения безопасности.

Добавьте на конвейер точку входа типа Audit. При создании выберите необходимые уровни угрозы для фильтрации входящего потока событий аудита.

Шаг 2. Маршрутизация данных

Чтобы направлять поступающие данные по разным путям обработки, необходимо добавить на конвейер маршрутизатор.

Маршрутизатор — это элемент конвейера, который позволяет на основе заданных фильтров направлять события по разным маршрутам в зависимости от выполнения условий фильтрации. Имеет один вход и несколько выходов: по одному выходу на каждый настроенный маршрут.

1. Добавьте на конвейер маршрутизатор. При добавлении элемента настройте необходимые маршруты. Например:

   Пример 17. Условие маршрута 1

   ```
   .meta.action_type == "Создание"
   ```

   Пример 18. Условие маршрута 2

   ```
   .meta.action_type == "Изменение"
   ```

   Пример 19. Условие маршрута 3

   ```
   .meta.action_type == "Удаление"
   ```

   Данные условия проверяют поле `action_type`, в котором хранится тип действия, породившего событие, и на основе его значения направляют событие по одному из настроенных маршрутов. В представленном примере учитываются действия типа "Создание", "Изменение" и "Удаление".

   В настройках маршрутизатора установите флажок **Добавить маршрут по умолчанию**, чтобы направлять события, не удовлетворяющие ни одному из заданных условий, в маршрут **По умолчанию**.

2. Выход из элемента **Точка входа** соедините со входом элемента **Маршрутизатор**.

Шаг 3. Конечные точки

Конечная точка пересылает событие в хранилище событий. Имеет один вход и не имеет выходов.

1. Добавьте на конвейер конечные точки, в которые будут сохраняться события. При добавлении

элементов выберите необходимые хранилища, куда будут направляться события требуемого типа.

Количество добавленных конечных точек должно совпадать с количеством маршрутов элемента **Маршрутизатор**.

2. Выходы из маршрутов элемента **Маршрутизатор** соедините со входами элементов **Конечная точка**.

Шаг 4. Включение коллектора и установка конфигурации конвейера

Перед включением конвейера убедитесь, что связанный с ним коллектор уже включен.

Чтобы события аудита начали поступать на конвейер и обрабатываться:

1. Перейдите в карточку коллектора и включите его.
2. Откройте конфигурацию конвейера и установите текущую конфигурацию.

   События, порождаемые действиями в системе, будут направляться в разные хранилища в зависимости от типа действия.

Мониторинг RQL-запросов и управление ресурсами

Сбор и хранение данных об RQL-запросах осуществляется в разделе **Диспетчер запросов**. В этом разделе отображается информация о запросах из разделов **Поиск**, **Дашборды**, времени их выполнения и объеме памяти, выделенной для выполнения. Здесь также можно управлять лимитами памяти, выделяемой для запросов.

По умолчанию в **Диспетчере** хранится информация о запросах, сделанных в течение 7 последних дней. Изменить срок хранения можно с помощью переменной окружения.

Интерфейс раздела

Вкладка Журнал запросов

В верхней части вкладки расположена панель инструментов, которая включает следующие компоненты:

- Кнопка экспорта ( ) открывает окно для экспорта истории запросов.
- Кнопка **Фильтр** ( ) позволяет задать критерии, по которым будут отобраны запросы.

В рабочей области отображается таблица найденных вхождений. Таблица содержит следующие столбцы:

- **Время запуска** — дата и время начала выполнения.
- **Статус** — состояние обработки запроса: **Запущен**, **Выполнен** или **Ошибка**.
- **Запустил** — имя учетной записи пользователя, который сделал запрос. При наведении отображается окно с именем и ролью пользователя.

- **Тип запроса** — раздел, где был сделан запрос: **Поиск**, **Дашборды**, **Общая статистика**, **Статистика по полю**.
- **Время выполнения** — время выполнения в миллисекундах.
- **Выделенная память** — память, выделенная для выполнения запроса, в мегабайтах с точностью до второго знака после десятичной точки.
- **Время окончания** — дата и время окончания (заполнено, если запрос завершен). Поле скрыто по умолчанию.
- **Теги** — пользовательские теги, указываются в карточке запроса.
- **Запрос RQL** — первые 20 символов запроса на RQL. При наведении курсора мыши отображается окно с полным запросом и кнопка для его копирования.
- **Запрос SQL** — первые 20 символов итогового запроса на SQL с учетом примененных фильтров, ограничений по времени и пагинации. При наведении курсора мыши отображается окно с полным запросом и кнопка для его копирования. Поле скрыто по умолчанию.
- **Ошибка** — описание ошибки (заполнено, если при выполнении запроса произошла ошибка). Поле скрыто по умолчанию.

При работе с таблицей доступны следующие операции:

- Фильтрация списка запросов. В параметрах фильтрации доступны следующие опции:
  - **Время запуска** — временной диапазон, в который был сделан запрос. Начало и конец диапазона задаются с помощью панели выбора даты и времени.
  - **Статус** — состояние выполнения, выбирается из выпадающего списка.
  - **Запустил** — пользователь, который запустил запрос, выбирается из выпадающего списка.
  - **Тип запроса** — раздел, где сделан запрос, выбирается из выпадающего списка.
  - **Минимальное время выполнения** — отображает запросы с длительностью выполнения, равной или большей заданному времени, указывается вручную в миллисекундах.
  - **Выделенная память** — отображает запросы, для которых выделено количество памяти, равное или большее заданного, указывается вручную в мегабайтах.
  - **Теги** вводятся вручную и применяются по нажатию клавиши ENTER.
- Сортировка по дате и времени начала, времени выполнения запроса и выделенной памяти.

  По умолчанию таблица упорядочена по времени начала выполнения от самых новых запросов к более старым.

- Обновление таблицы.
- Настройка отображения таблицы.

При выборе отдельного запроса в правой части рабочей области отображается карточка с детальной информацией о нем.

Вкладка Управление ресурсами

В верхней части вкладки расположена панель инструментов, которая включает следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новый лимит памяти.
- Поле поиска позволяет искать лимиты по названию.
- Кнопка **Фильтр** ( ) позволяет задать критерии, по которым будут отобраны лимиты.

В рабочей области отображается таблица найденных вхождений. Таблица содержит следующие столбцы:

- **Название** — имя, используемое для идентификации лимита в системе.
- **Статус** — состояние лимита (включен/выключен).
- **Роль** — список ролей, для которых установлен лимит.
- **Лимит** — лимит выделяемой памяти в мегабайтах.
- **Дата создания** — дата и время создания лимита.
- **Дата изменения** — дата и время изменения лимита.

При работе с таблицей доступны следующие операции:

- Фильтрация списка лимитов по ролям, для которых установлены лимиты.
- Сортировка таблицы по названию, объему, датам создания и изменения лимита.
- Настройка отображения таблицы.

При выборе отдельного лимита в правой части рабочей области отображается карточка с детальной информацией о нем.

Просмотр запроса

Чтобы просмотреть сведения о запросе, нажмите на строку запроса в списке. В правой части экрана отобразится карточка с подробной информацией.

В верхней части карточки отображается идентификатор запроса.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка содержит следующие поля:

- **Время запуска** — дата и время начала выполнения.
- **Статус** — состояние обработки запроса: **Запущен**, **Выполнен** или **Ошибка**.
- **Ошибка** — сообщение об ошибке (поле заполнено, если запрос завершился с ошибкой).
- **Запустил** — имя и роль учетной записи пользователя, который сделал запрос.
- **Время выполнения** — время выполнения с точностью до миллисекунд.
- **Выделенная память** — память, выделенная для выполнения запроса, в мегабайтах.
- **Тип запроса** — раздел, где был сделан запрос: **Поиск**, **Дашборды**, **Общая статистика**, **Статистика по полю**.
- **Время окончания** — дата и время окончания (поле заполнено, если запрос завершен).

- **Теги** — теги запроса, указываются вручную.
- **Запрос RQL** — запрос на языке RQL и кнопка для его копирования.
- **Запрос SQL** — запрос на SQL и кнопка для его копирования.

Экспорт истории запросов

Чтобы экспортировать историю запросов:

1. Нажмите на кнопку **Экспорт** ( ) на панели инструментов. Отобразится окно экспорта истории.
2. Нажмите на кнопку **Экспортировать**. Система загрузит историю запросов на устройство пользователя в формате CSV и отобразит соответствующее уведомление.

Работа с лимитами

Лимиты регулируют объем потребления оперативной памяти для определенных ролей пользователей. Лимит можно установить для нескольких ролей, но для каждой роли допускается только один лимит.

Лимиты применяются в следующем порядке:

- При выполнении запроса система проверяет лимит роли пользователя.
- Если у пользователя есть несколько ролей, для которых установлены лимиты, используется наибольший из лимитов.
- Если для ролей пользователя нет включенных лимитов, система использует ограничение на максимальное потребление памяти из настроек ClickHouse.

Доступные операции над лимитами:

- Создание лимита
- Просмотр лимита
- Включение и выключение лимита
- Изменение лимита
- Удаление лимита

Создание лимита

Создавать лимиты могут администраторы, инженеры, пользователи, которые имеют соответствующее разрешение в составе своих ролей, а также пользователи со статусом **Суперадминистратор**.

Чтобы создать лимит:

1. Перейдите в раздел **Инструменты** → **Диспетчер запросов** и откройте вкладку **Управление ресурсами**.
2. Нажмите на кнопку **Создать** ( ). Отобразится окно создания лимита.
3. Введите название лимита.

4. Выберите из выпадающего списка роли, для которых устанавливается лимит.
5. Введите значение лимита в мегабайтах (по умолчанию — 1024 Мб).
6. Нажмите на кнопку **Создать**. Система создаст лимит и отобразит соответствующее уведомление. Новый лимит появится в разделе **Инструменты** → **Диспетчер запросов** на вкладке **Управление ресурсами**.

Если для каких-либо из указанных ролей уже заданы лимиты, система отобразит уведомление об ошибке.

Просмотр лимита

Чтобы просмотреть лимит:

1. Перейдите в раздел **Инструменты** → **Диспетчер запросов** и откройте вкладку **Управление ресурсами**. Система отобразит сведения об имеющихся лимитах.
2. Нажмите на строку лимита в списке. Система отобразит в правой части экрана карточку с подробной информацией о лимите.

В верхней части карточки отображается название лимита.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка содержит следующие элементы:

- Переключатель статуса лимита.
- **ID** — идентификатор лимита.
- **Название** — имя, используемое для идентификации лимита в системе.
- **ID тенанта** — идентификатор тенанта.
- **Статус** — состояние лимита (включен/выключен).
- **Роль** — роли, для которых установлен лимит.
- **Лимит** — лимит выделяемой памяти в мегабайтах.
- **Дата создания** — дата и время создания лимита.
- **Создал** — имя учетной записи пользователя, который создал лимит.
- **Дата изменения** — дата и время изменения лимита.
- **Изменил** — имя учетной записи пользователя, который изменил лимит.

Включение и выключение лимита

Чтобы лимит учитывался при выполнении RQL-запросов, его нужно включить.

Чтобы включить лимит:

1. Перейдите в раздел **Инструменты** → **Диспетчер запросов** и откройте вкладку **Управление ресурсами**. Система отобразит сведения об имеющихся лимитах.

2. Нажмите на строку лимита в списке. Система отобразит в правой части экрана карточку с подробной информацией об этом лимите.
3. Переведите переключатель состояния лимита в верхней части карточки в активное положение. Отобразится окно подтверждения.
4. Нажмите на кнопку **Включить**. Система включит лимит и отобразит соответствующее уведомление. Лимит станет действовать при выполнении запросов.

Чтобы выключить действующий лимит:

1. Перейдите в раздел **Инструменты** → **Диспетчер запросов** и откройте вкладку **Управление ресурсами**. Система отобразит сведения об имеющихся лимитах.
2. Нажмите на строку лимита в списке. Система отобразит в правой части экрана карточку с подробной информацией об этом лимите.
3. Переведите переключатель в верхней части карточки в неактивное положение. Отобразится окно подтверждения.
4. Нажмите на кнопку **Выключить**. Система выключит лимит и отобразит соответствующее уведомление. Лимит перестанет действовать при выполнении запросов.

Изменение лимита

Чтобы изменить лимит:

1. Перейдите в раздел **Инструменты** → **Диспетчер запросов** и откройте вкладку **Управление ресурсами**. Система отобразит сведения об имеющихся лимитах.
2. Нажмите на строку лимита в списке. Система отобразит в правой части экрана карточку с подробной информацией об этом лимите.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек лимита.
4. Внесите изменения в требуемые поля.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке и списке лимитов.

Название лимита также можно изменить непосредственно из его карточки:

1. Перейдите в раздел **Инструменты** → **Диспетчер запросов** и откройте вкладку **Управление ресурсами**. Система отобразит сведения об имеющихся лимитах.
2. Нажмите на строку лимита в списке. Система отобразит в правой части экрана карточку с подробной информацией об этом лимите.
3. Нажмите на кнопку рядом с полем **Название** в карточке. Система отобразит поле ввода значения.
4. Введите новое название.
5. Нажмите **Сохранить**. Система сохранит измененное название и отобразит соответствующее уведомление. Новое название отобразится в карточке и списке лимитов.

Удаление лимита

Чтобы удалить лимит:

1. Перейдите в раздел **Инструменты** → **Диспетчер запросов** и откройте вкладку **Управление ресурсами**. Система отобразит сведения об имеющихся лимитах.
2. Нажмите на строку лимита в списке. Система отобразит в правой части экрана карточку с подробной информацией об этом лимите. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно для подтверждения удаления.
3. Нажмите на кнопку **Удалить**. Система удалит лимит и отобразит соответствующее уведомление. Удаленный лимит будет исключен из списка. Лимит перестанет действовать при выполнении запросов.

Устранение проблем при эксплуатации системы

В данном разделе приведены возможные проблемы при работе системы и способы их решения.

ClickHouse: не выполняются распределенные запросы (`on cluster`)

Версия системы: 3.0 и выше.

Проявление: в графическом интерфейсе не создаются новые объекты, такие как хранилища или активные списки.

В консоли кластера ClickHouse не выполняются распределенные запросы, содержащие секцию `on cluster`. Например, не создается распределенная таблица `test`:

```
create table test on cluster '{cluster}' (id UUID) Engine = MergeTree order by id
```

Причина: сбой запроса, который привел к невозможности обработки очереди запросов.

Решение:

1. Получите список подов, на которых расположены шарды ClickHouse:

   ```
   kubectl -n <namespace> get pods -l app.kubernetes.io/name=clickhouse
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.
2. Найдите в логах шардов ошибку очереди запросов:

   ```
   kubectl logs -n <namespace> <shard_name> --all-containers=true | grep
   "DB::Exception: Cannot create status dirs"
   ```

   Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<shard_name>` — имя шарда из списка, полученного на шаге 1, например, `clickhouse-shard1-0`.

  Пример ошибки очереди запросов:

  ```
  2024.12.12 10:31:51.883015 [ 410 ] {} <Error> DDLWorker: Unexpected error (1559
  times in a row), will try to restart main thread: Code: 341. DB::Exception:
  Cannot create status dirs for /clickhouse/task_queue/ddl/query-0007458048, most
  likely because someone is deleting it concurrently. (UNFINISHED), Stack trace
  (when copying this message, always include the lines below)
  ```

3. Сохраните идентификатор запроса в очереди, вызвавшего ошибку: в примере выше — `/clickhouse/task_queue/ddl/query-0007458048`.
4. Проверьте размер очереди запросов ClickHouse на шарде, на котором возникла ошибка:

   ```
   kubectl exec -n <namespace> <clickhouse_shard_name> — clickhouse-keeper-client -h
   127.0.0.1 -p 2181 -q "ls 'clickhouse/task_queue/ddl'" 2>/dev/null | tr ' ' "\n"| wc
   -l
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.
   - `<clickhouse_shard_name>` — имя шарда ClickHouse, на котором возникла ошибка, например, `clickhouse-shard1-0`.
5. Удалите запрос из очереди ClickHouse:

   ```
   kubectl exec -n <namespace> <clickhouse_shard_name> -it — clickhouse-keeper-client
   -h 127.0.0.1 -p 2181 -q "rmr '<query_id>'"
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.
   - `<clickhouse_shard_name>` — имя шарда ClickHouse, на котором возникла ошибка, например, `clickhouse-shard1-0`;
   - `<query_id>` — идентификатор запроса из сообщения об ошибке, полученный на шаге 3.
6. Повторно проверьте размер очереди запросов ClickHouse, выполнив команду из шага 4. Убедитесь, что размер очереди уменьшается, что подтверждает возобновление ее обработки.
7. Повторно выполните распределенный запрос.

ClickHouse: ошибка аутентификации при выполнении запросов

Версия системы: 3.0 и выше.

Проявление: запросы к БД ClickHouse завершаются с ошибкой `Authentication failed: password is`

`incorrect, or there is no user with such name`. Это может происходить в любом разделе, где осуществляется работа с событиями, например, в разделе **Поиск**.

Причина: не совпадают пароли пользователей `owner`, `reader` или `writer` в объектном хранилище NATS Key/Value и в конфигурации ClickHouse.

Решение: установите единые пароли в объектном хранилище и в конфигурации ClickHouse. Для этого:

1. Получите пароли пользователей из NATS Key/Value:
   a. Подключитесь к терминалу управляющего узла кластера.
   b. Подключитесь к терминалу пода `nats-main-box`, выполнив следующую команду:

      ```
      kubectl exec -itn <namespace> deployment/nats-main-box -c nats-box -- sh
      ```

      Здесь:

      - `<namespace>` — имя пространства имен, в котором установлен кластер.

   c. Получите пути к ключам, в которых хранятся пароли ClickHouse, выполнив следующую команду:

      ```
      nats kv ls siem
      ```

      ▼ *Пример путей к ключам, в которых хранятся пароли ClickHouse*

      ```
      space-main/evo-clickhouse1/owner
      space-main/evo-clickhouse1/reader
      space-main/evo-clickhouse1/writer
      ```

   d. Просмотрите пароли пользователей ClickHouse, выполнив для каждого из них команду:

      ```
      nats kv get siem <path_to_key> --raw
      ```

      Здесь:

      - `<path_to_key>` — путь к ключу, где хранится пароль.

2. Определите, пароль какого пользователя в конфигурации ClickHouse не совпадает с паролем в хранилище. Для этого:
   a. Осуществите проброс портов ClickHouse, выполнив на хосте с сетевым доступом к управляющему узлу кластера следующую команду:

      ```
      kubectl port-forward -n <namespace> service/clickhouse <local_click_port>:8123
      ```

      Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<local_click_port>` — порт на хосте, по которому будет осуществляться доступ к ClickHouse.

b. В браузере откройте URL `localhost:<local_click_port>/play`, где `<local_click_port>` — порт с предыдущего шага. Отобразится графический клиент ClickHouse.

c. Выполните следующие действия для каждого из пользователей `owner`, `reader` и `writer`:

i. Введите логин и пароль в поля в правом верхнем углу экрана.

ii. Выполните любой запрос, например, `SELECT * FROM default.disks`.

Если возникает ошибка `Authentication failed: password is incorrect, or there is no user with such name. (AUTHENTICATION_FAILED)`, то пароль текущего пользователя необходимо исправить в таблице пользователей ClickHouse.

3. Получите список подов, на которых расположены шарды ClickHouse:

```
kubectl -n <namespace> get pods -l app.kubernetes.io/name=clickhouse
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.

4. Получите с любого шарда имя и пароль пользователя ClickHouse, чтобы впоследствии выполнить запросы к базе данных через консольный клиент:

a. Чтобы получить имя пользователя ClickHouse, выведите подробную информацию о поде шарда:

```
kubectl -n <namespace> describe pod <shard_pod>
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<shard_pod>` — имя пода, на котором расположен шард ClickHouse.

Имя пользователя ClickHouse расположено в разделе `Containers → <имя контейнера ClickHouse> → Environment → CLICKHOUSE_ADMIN_USER`.

b. Получите пароль пользователя ClickHouse из секрета `clickhouse` и декодируйте его из base64:

```
kubectl -n <namespace> get secret clickhouse -o jsonpath='{.data.admin-
password}' | base64 -d
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.

5. Для каждого пользователя, для которого возникла ошибка аутентификации, обновите пароль,

выполнив следующую команду на каждом шарде:

```
kubectl exec -n <namespace> <shard_pod> -- clickhouse-client --user
<clickhouse_user> --password <clickhouse_password> "ALTER USER <user_to_edit>
IDENTIFIED BY '<password_from_storage>'"
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<shard_pod>` — имя пода, на котором расположен шард ClickHouse.
- `<clickhouse_user>` — имя пользователя ClickHouse, полученное на шаге 4.
- `<clickhouse_password>` — пароль пользователя ClickHouse, полученный на шаге 4.
- `<user_to_edit>` — пользователь, пароль которого нужно изменить: `owner`, `reader` или `writer`.
- `<password_from_storage>` — пароль пользователя из объектного хранилища, полученный на шаге 1.

6. Для пользователя `writer` также следует обновить пароль в карте конфигурации `clickhouse`. Чтобы открыть ее, выполните следующую команду:

```
kubectl -n <namespace> edit configmap clickhouse
```

Укажите пароль пользователя `writer` из объектного хранилища как значение ключа `remote_servers → default → shard → replica → password`.

Ключей `remote_servers → default → shard → replica → password` столько же, сколько шардов. Необходимо установить пароль из объектного хранилища для каждого шарда.

7. Удалите поды ClickHouse. Они будут пересозданы автоматически и начнут использовать новые пароли:

```
kubectl -n <namespace> delete pods -l app.kubernetes.io/name=clickhouse
```

Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.

Работа с событиями: ошибки в логах сервисов

Версия системы: 3.0 и выше. Должен быть установлен провайдер хранилища Longhorn. Инсталляция ClickHouse находится внутри кластера.

Проявление: при работе системы с потоком событий в логах сервисов возникает ошибка, содержащая строку `EXT4-fs error (device sdu)`.

Причина: конфигурация оборудования не соответствует техническим требованиям системы.

Например, выделено мало ядер CPU.

Решение: приведите конфигурацию оборудования в соответствие с техническими требованиями системы.

Работа с событиями: не работает поиск, не строятся графики на дашбордах

Версия системы: 3.0 и выше.

Проявление: не работает поиск, не строятся графики на дашбордах.

При выполнении поискового запроса в консоли браузера возникает ошибка вида:

```
search::search::endpoints::simplified_search: error: Invalid state: Unable get
connection info: Unable to get connection info for event storage: grpc status:
code=13, message=error trying to connect: tcp open error: Address family not supported
by protocol (os error 97)
```

Причина:

- либо gRPC-порт инсталляции ClickHouse недоступен или не совпадает с указанным при установке системы, поэтому сервис поиска `search` не может подключиться к ClickHouse;
- либо сервис поиска не может подключиться к сервису `event-storage-manager`.

Решение:

1. Получите номер gRPC-порта из значения ключа `grpc_port` конфигурации ClickHouse.
2. Попробуйте выполнить поиск или построить график несколько раз, чтобы в логах пода накопились сообщения об ошибках.
3. Просмотрите логи пода `event-storage-manager`:
   a. Получите полное имя пода сервиса `event-storage-manager`, выполнив следующую команду:

   ```
   kubectl -n <namespace> get pods -l app.kubernetes.io/name=evo.siem.event-
   storage-manager
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.

   ▼ *Пример вывода имени пода* `event-storage-manager`

   ```
   NAME                                              READY   STATUS
   RESTARTS   AGE
   evo.siem.event-storage-manager-7db4bc4c99-wlqlk   1/1     Running   0
   21m
   ```

   В примере выше полное имя пода `event-storage-manager` — `evo.siem.event-storage-`

manager-7db4bc4c99-wlqlk.

b. Для просмотра логов пода event-storage-manager используйте руководство **Просмотр логов пода**. В качестве имени пода (<pod_name>) укажите полное имя пода event-storage-manager, полученное на предыдущем шаге.

4. В логах пода может присутствовать номер gRPC-порта ClickHouse, используемого сервисом event-storage-manager, как значение ключа clickHouseConnectionInfo → port. В примере ниже — 9100:

```
{"level":10,"time":1738668412977,"pid":1,"hostname":"evo.siem.event-storage-
manager-66667479f5-
74tn2","context":"EventStorageService","clickHouseConnectionInfo":{"host":"clickhou
se","password":"REDACTED","port":9100,"user":"owner"},"requestId":"[missing]","msg"
:"connection info"}
```

a. Если ключ clickHouseConnectionInfo → port присутствует в логах и номер порта отличается от номера в конфигурации ClickHouse:

i. Укажите номер порта из clickHouseConnectionInfo → port как значение ключа grpc_port конфигурации ClickHouse.

ii. Удалите поды ClickHouse. Они будут пересозданы автоматически и будут использовать новый gRPC-порт:

```
kubectl -n <namespace> delete pods -l app.kubernetes.io/name=clickhouse
```

Здесь:

- <namespace> — имя пространства имен, в котором установлен кластер.

b. Если ключ clickHouseConnectionInfo → port отсутствует или номера портов в конфигурации ClickHouse и логах совпадают, удалите поды сервисов search и bff:

```
kubectl -n <namespace> delete pods -l app.kubernetes.io/name=evo.siem.search
kubectl -n <namespace> delete pods -l app.kubernetes.io/name=evo.core.bff
```

Здесь:

- <namespace> — имя пространства имен, в котором установлен кластер.

Удаленные поды будут автоматически пересозданы в течение нескольких минут. Когда поды будут иметь статус Running, можно будет повторить поиск или построение графика.

Если после пересоздания подов поиск и графики на дашбордах не работают, обратитесь в службу поддержки.

WebSocket: ошибки запросов

Версия системы: 3.0 и выше.

Проявление: не работают функции, использующие протокол WebSocket: например, поиск, импорт элементов экспертизы или скачивание отчетов. В консоли браузера запросы по протоколу WebSocket завершаются с ошибками.

Причина: для сервиса API `evo.siem.bff` не включены подписки WebSocket, или есть проблема с соединениями WebSocket на уровне инфраструктуры.

Решение:

1. Включите подписки WebSocket:
   a. Откройте карту конфигурации `evo.siem-bff-env`, выполнив следующую команду:

      ```
      kubectl -n <namespace> edit configmap evo.siem-bff-env
      ```

      Здесь:

      - `<namespace>` — имя пространства имен, в котором установлен кластер.

   b. Если переменная `SIEM_BFF_GRAPHQL_ENABLE_PLAYGROUND` имеет значение `"false"`, включите подписки WebSocket:
      i. Установите для переменной `SIEM_BFF_GRAPHQL_SUBSCRIPTION_TRANSPORT_WS_ENABLED` значение `"true"`.
      ii. Закройте карту конфигурации.
      iii. Удалите под сервиса `evo.siem.bff`, выполнив следующую команду:

      ```
      kubectl -n <namespace> delete pods -l app.kubernetes.io/name=evo.siem.bff
      ```

      Под будет пересоздан со включенными подписками WebSocket.

2. Если включение подписок не решило проблему, или переменная `SIEM_BFF_GRAPHQL_ENABLE_PLAYGROUND` уже имеет значение `"true"`, выполните запрос с помощью любой утилиты wss, например, wscat, к следующему URL:

   ```
   wss://<system_url>/api/graphql
   ```

   Здесь:

   - `<system_url>` — URL, по которому осуществляется доступ к системе.

   Если попытка установить WSS-соединение завершается с ошибкой, то вероятная причина проблемы — нарушения в инфраструктуре, в которой установлена система: например, сбой сети, некорректные настройки прокси или ограничение файрвола.

Коллекторы: падение или отсутствие EPS

Версия системы: 3.0 и выше.

Проявление: значительное падение EPS (количества событий в секунду) или отсутствие событий на коллекторе.

Ошибки в логе коллектора:

```
"line": "2024-11-28T11:39:52.739224Z ERROR source{component_kind=\"source\"
component_id=4b5ae224-78fc-4c00-9614-5ecb41871f2b_eJ__wBp4vtuHe9YVoOxCm_syslog
component_type=socket}: vector::internal_events::codecs: Failed framing bytes.
error=Unable to decode input as UTF8 error_code=\"decoder_frame\"
error_type=\"parser_failed\" stage=\"processing\" internal_log_rate_limit=true",
"line": "2024-11-28T11:39:52.753215Z ERROR source{component_kind=\"source\"
component_id=4b5ae224-78fc-4c00-9614-5ecb41871f2b_eJ__wBp4vtuHe9YVoOxCm_syslog
component_type=socket}: vector::topology: An error occurred that Vector couldn't
handle: the task panicked and was aborted.",
```

Причина: выбран некорректный тип точки входа, не соответствующий формату поступающих событий; например, в точку входа типа Syslog поступают события в формате JSON.

Решение: необходимо проверить конфигурацию точек входа конвейеров, установленных в коллекторе, и привести их типы в соответствие с форматом поступающих событий.

Коллекторы: не отображаются метрики коллекторов, конвейеров и элементов конвейеров

Версия системы: 3.0 и выше.

Проявление: в карточках коллекторов и конвейеров, а также на элементах конвейеров в режиме активной версии не отображаются метрики.

Причина: поду сервиса сбора метрик Prometheus не хватает размера тома (Persistent Volume Claim) для хранения данных.

Решение: нужно увеличить PVC для Prometheus. Для этого:

1. Получите имя PVC Prometheus, выполнив следующую команду:

   ```
   kubectl -n kube-prometheus-stack describe pod prometheus-kube-prometheus-stack-
   prometheus-0
   ```

   Имя PVC Prometheus является значением ключа `Volumes → <prometheus_volume_name> → ClaimName`.

   ▼ *Пример имени PVC Prometheus*

   ```
   # ...
   Volumes:
     prometheus-kube-prometheus-stack-prometheus-db:
       Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in
   the same namespace)
   ```

```
    ClaimName:  prometheus-kube-prometheus-stack-prometheus-db-prometheus-kube-
prometheus-stack-prometheus-0
    ReadOnly:   false
# ...
```

В этом случае имя PVC Prometheus — `prometheus-kube-prometheus-stack-prometheus-db-prometheus-kube-prometheus-stack-prometheus-0`.

2. Откройте конфигурацию PVC для Prometheus, выполнив следующую команду:

```
kubectl -n kube-prometheus-stack edit pvc <prometheus_pvc_name>
```

Здесь:

- `<prometheus_pvc_name>` — имя PVC Prometheus, полученное на предыдущем шаге.

3. Увеличьте размер PVC, отредактировав значение ключа `spec → resources → requests → storage`. Формат и единицы измерения размеров PVC приведены в документации Kubernetes.

```
# ...
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      # Отредактируйте значение этого ключа.
      storage: 15Gi
# ...
```

4. Закройте файл. Kubernetes перераспределит дисковое пространство в соответствии с обновленным PVC, и размер тома изменится через некоторое время.

Точка входа типа Database (MS SQL): не поступают события

Версия системы: 3.0 и выше.

Проявление: не поступают события в точку входа типа Database с драйвером MS SQL.

Ошибки в логе коллектора:

```
ERROR vector::sources::database: JDBC Server unexpected stopped error=Unable to open
JDBC Connection
PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException:
unable to find valid certification path to requested target.
```

Причина: ошибка при разборе сертификата сервера БД, что приводит к невозможности установить защищенное SSL-соединение.

Решение: отключите SSL для соединения с БД. Для этого измените настройки точки входа, добавив в **Адрес подключения** параметры `trustServerCertificate=false` и `encrypt=false`.

Пример строки подключения:

```
jdbc:sqlserver://localhost;user=user;password=password;trustServerCertificate=false;en
crypt=false
```

Если подобная ошибка возникает при использовании другой СУБД, кроме MS SQL, получите информацию о способах отключения SSL в официальной документации JDBC-драйвера соответствующей СУБД.

Базы данных событий: не удается зарегистрировать новую базу данных

Версия системы: 3.0 и выше. ClickHouse установлен на выделенном сервере (вынесенная установка).

Проявление: в интерфейсе системы не удается зарегистрировать новую базу данных событий. В логах сервера ClickHouse возникает ошибка вида:

```
Error when deleting table of event storage: All connection tries failed while
connecting to ZooKeeper. nodes:
[fe80::20c:29ff:fe89:fa80%ens33]:9181\nPoco::Exception. Code: 1000, e.code() = 111,
Connection refused (version 24.3.2.23 (official build)),
[fe80::20c:29ff:fe89:fa80%ens33]:9181\n
```

Причина: если во время установки системы выбрана вынесенная установка ClickHouse и указано полное доменное имя (FQDN) сервера ClickHouse, то при выполнении запроса доменное имя по умолчанию преобразуется в адрес IPv6. Это приводит к ошибке, если у сервера ClickHouse нет IPv6-адреса.

Решение: следует отключить использование IPv6 на сервере, на котором установлен ClickHouse. Для этого выполните следующие действия на сервере, на котором установлен ClickHouse:

1. Откройте файл /etc/sysctl.conf.
2. Добавьте в файл следующие строки:

   ```
   net.ipv6.conf.all.disable_ipv6 = 1
   net.ipv6.conf.default.disable_ipv6 = 1
   net.ipv6.conf.local.disable_ipv6 = 1
   ```

3. Сохраните файл.
4. Примените изменения с помощью следующей команды:

   ```
   sysctl -p
   ```

5. Перезагрузите службу ClickHouse:

```
sudo service clickhouse-server restart
```

Хранилища событий: не отображаются события за последние несколько минут

Версия системы: 3.0 и выше.

Проявление: при просмотре хранилища событий или поиске по нему не отображаются события, поступившие в систему за последние несколько минут.

Причина: отставание времени на узлах от текущего. Получить время на узлах можно, выполнив в их консоли команду `date`.

Решение: настройте синхронизацию времени на узлах согласно документации операционной системы, которая установлена на этих узлах. Например, документация по синхронизации времени в Astra Linux расположена на справочном портале разработчика ОС.

Хранилища событий аудита: не выполняется поиск или не поступают события

Версия системы: 3.0 и выше.

Проявление: возникает внутренняя ошибка сервера при поиске в хранилище событий аудита или в хранилище событий аудита не поступают новые события.

Причина: на некоторых шардах ClickHouse отсутствует таблица, соответствующая хранилищу событий аудита.

Решение:

1. Получите ID хранилища событий (далее `<TABLE_ID>`) из его карточки.
2. Получите список подов, на которых расположены шарды ClickHouse:

   ```
   kubectl -n <namespace> get pods -l app.kubernetes.io/name=clickhouse
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.
3. Получите с любого шарда имя и пароль пользователя ClickHouse, чтобы впоследствии выполнить запросы к базе данных через консольный клиент:
   a. Чтобы получить имя пользователя ClickHouse, выведите подробную информацию о поде шарда:

      ```
      kubectl -n <namespace> describe pod <shard_pod>
      ```

      Здесь:

- `<namespace>` — имя пространства имен, в котором установлен кластер.
- `<shard_pod>` — имя пода, на котором расположен шард ClickHouse.

  Имя пользователя ClickHouse расположено в разделе `Containers → <имя контейнера ClickHouse> → Environment → CLICKHOUSE_ADMIN_USER`.

b. Получите пароль пользователя ClickHouse из секрета `clickhouse`:

   i. Получите значение секрета:

   ```
   kubectl get secret clickhouse -o jsonpath='{.data}'
   ```

   ▼ *Пример вывода значения секрета*

   ```
   {"admin-password":"ABCDEF=="}
   ```

   ii. Декодируйте значение секрета из base64. Результат декодирования — пароль пользователя ClickHouse.

   ```
   echo <encoded_secret> | base64 --decode
   ```

   Здесь:

   - `<encoded_secret>` — закодированное значение `admin-password` с предыдущего шага, например, `ABCDEF==`.

4. На каждом шарде ClickHouse проверьте, существуют ли таблицы `EventStorage_<TABLE_ID>` и `EventStorage_<TABLE_ID>_local`. Если они отсутствуют, создайте их. Для этого:

   a. Выполните следующие запросы:

   ```
   kubectl exec -n <namespace> <shard_pod> -- clickhouse-client --user
   <clickhouse_user> --password <clickhouse_password> 'SELECT 1 FROM
   "EventStorage_<TABLE_ID>" LIMIT 1'
   ```

   ```
   kubectl exec -n <namespace> <shard_pod> -- clickhouse-client --user
   <clickhouse_user> --password <clickhouse_password> 'SELECT 1 FROM
   "EventStorage_<TABLE_ID>_local" LIMIT 1'
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.
   - `<shard_pod>` — имя пода, на котором расположен шард ClickHouse.
   - `<clickhouse_user>` — имя пользователя ClickHouse.
   - `<clickhouse_password>` — пароль пользователя ClickHouse.

b. Изучите результат выполнения запросов:

- Если в результате запроса отображается 1, значит, таблица существует на шарде. В таком случае получите запрос для ее создания, выполнив следующий запрос и скопировав его вывод:

```
kubectl exec -n <namespace> <shard_pod> -- clickhouse-client --user
<clickhouse_user> --password <clickhouse_password> 'SHOW CREATE TABLE
<table_name>'
```

Здесь:

- <namespace> — имя пространства имен, в котором установлен кластер.
- <shard_pod> — имя пода, на котором расположен шард ClickHouse.
- <clickhouse_user> — имя пользователя ClickHouse.
- <clickhouse_password> — пароль пользователя ClickHouse.
- <table_name> — имя таблицы.

Пример 20. Пример вывода запроса для создания таблицы

```
CREATE TABLE
default.EventStorage_91bbadf5_0c77_4719_b4d0_dc6b8f298cab\n(\n    `id`
UUID,\n    `sourceIp` IPv6,\n    `tenantId` LowCardinality(String),\n
`art` DateTime,\n    `raw` String,\n    `collectorId`
LowCardinality(String),\n    `timestamp` DateTime,\n    `type`
Enum8(\'Unspecified\' = -1, \'исходное событие\' = 0, \'нормализованное
событие\' = 1, \'агрегированное событие\' = 2, \'корреляционное событие\'
= 3, \'событие аудита\' = 4),\n    `dvc` IPv6,\n    `rt` DateTime,\n
`suser` LowCardinality(String),\n    `eventType`
LowCardinality(String),\n    `moduleKey` LowCardinality(String),\n
`eventRiskLevel` Enum8(\'Unspecified\' = -1, \'DEBUG\' = 1, \'LOW\' = 2,
\'MEDIUM\' = 3, \'HIGH\' = 4, \'CRITICAL\' = 5, \'FATAL\' = 6,
\'ACCIDENT\' = 7),\n    `data` String,\n    `meta` String\n)\nENGINE =
Distributed(\'{cluster}\', \'default\',
\'EventStorage_91bbadf5_0c77_4719_b4d0_dc6b8f298cab_local\', rand())
```

- Если в результате запроса отображается ошибка UNKNOWN_TABLE, значит, таблица на шарде отсутствует. Чтобы создать ее, используйте запрос, полученный с другого шарда:

```
kubectl exec -n <namespace> <shard_pod> -- clickhouse-client --user
<clickhouse_user> --password <clickhouse_password> '<create_query>'
```

Здесь:

- <namespace> — имя пространства имен, в котором установлен кластер.
- <shard_pod> — имя пода, на котором расположен шард ClickHouse.

- `<clickhouse_user>` — имя пользователя ClickHouse.
- `<clickhouse_password>` — пароль пользователя ClickHouse.
- `<create_query>` — запрос на создание таблицы, полученный с шарда, на котором она существует.

5. Получите имя пользователя PostgreSQL, выведя подробную информацию о поде PostgreSQL:

   ```
   kubectl -n <namespace> describe pod postgresql-0
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.

     Имя пользователя ClickHouse расположено в разделе `Containers` → `postgresql` → `Environment` → `POSTGRES_USER`.

6. Удалите запись о таблице, соответствующей хранилищу событий аудита, из таблицы `event_storage` БД `event_storage` PostgreSQL:

   ```
   kubectl exec -n <namespace> postgresql-0 -- psql -d event_storage
   --username=<postgresql_user> -c "DELETE FROM event_storage WHERE table_name =
   'EventStorage_<TABLE_ID>'"
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.
   - `<postgresql_user>` — имя пользователя PostgreSQL.

7. Удалите под `audit-manager`:

   ```
   kubectl -n <namespace> delete pods -l app.kubernetes.io/name=evo.siem.audit-manager
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.

     При пересоздании пода `audit-manager` таблица, соответствующая хранилищу событий аудита, будет создана заново.

Экспертиза: не удается импортировать элементы экспертизы

Версия системы: 3.0 и выше.

Проявление: импорт некоторых элементов экспертизы завершается с ошибкой `Worker terminated`.

Причина: сервис `evo.siem.pipeline-verification` обладает недостаточными ресурсами для полной обработки импорта.

Решение:

1. Подключитесь к кластеру с помощью Lens.
2. Увеличьте значения таймаутов для сервиса `evo.siem.pipeline-verification`. Для этого:
   a. В выпадающем меню свойств кластера выберите пункт **Config** → **Config Maps**.
   b. Выберите пространство имен, в котором установлен кластер, в выпадающем списке **Select Namespace** в правой части экрана.
   c. Выберите карту конфигурации **evo.siem.pipeline-verification-env** из списка.
   d. В открывшейся панели справа увеличьте значения следующих переменных среды:
      - `SIEM_PIPELINE_VERIFICATION_ROBJECT_RUN_TEST_TIMEOUT` — таймаут для выполнения тестов элемента экспертизы в миллисекундах. Значение по умолчанию: `60000`.
      - `SIEM_PIPELINE_VERIFICATION_ROBJECT_CHECK_SCHEMA_TIMEOUT` — таймаут для проверки конфигурации элемента экспертизы в миллисекундах. Значение по умолчанию: `60000`.
3. Увеличьте лимиты ресурсов для сервиса `evo.siem.pipeline-verification`. Для этого:
   a. В выпадающем меню свойств кластера выберите пункт **Workloads** → **Deployments**.
   b. Выберите пространство имен, в котором установлен кластер, в выпадающем списке **Select Namespace** в правой части экрана.
   c. Выберите сервис **evo.siem.pipeline-verification** из списка.
   d. Нажмите на кнопку редактирования в верхней части правой панели.
   e. В отобразившейся конфигурации увеличьте значения лимитов оперативной памяти и процессора в секции `resources`:

```
resources:
  limits:
    cpu: 1500m
    memory: 600Mi
  requests:
    cpu: 100m
    memory: 200Mi
```

   f. Нажмите на кнопку **Save**. Новая конфигурация будет применена.

Авторизация

Перед входом в систему запросите у системного администратора:

- адрес для входа в веб-интерфейс продукта;
- логин и пароль вашей учетной записи пользователя.

По умолчанию для первого входа в систему используются учетные данные **admin** /**admin**.

При переходе по адресу на экране отобразится страница авторизации, которая содержит поля для ввода логина и пароля учетной записи пользователя, а также выпадающий список для выбора языка интерфейса в системе.

Вход в систему

Чтобы выполнить вход в систему:

1. В адресной строке браузера введите ссылку для входа в веб-интерфейс системы. На экране отобразится страница авторизации.
2. Укажите логин и пароль учетной записи пользователя и нажмите на кнопку **Войти**.

Если в систему добавлены пользователи с таким же логином из других доменов, укажите домен в поле логина. В противном случае домен можно не указывать.

Если срок действия пароля истекает менее, чем через семь дней или пароль не соответствует текущим требованиям, система предоставит ссылку для его смены.

Если вход в систему выполнен успешно, в окне браузера отобразится стартовый раздел системы. В правом верхнем углу экрана отобразится текущая учетная запись.

Выход из системы

Чтобы выйти из системы:

1. Откройте выпадающее меню вашей учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Выйти из системы** ([→). Система завершит сеанс, и на экране отобразится страница авторизации.

Интерфейс

После успешной авторизации в браузере отобразится окно стартового раздела веб-интерфейса системы.

В верхней правой части интерфейса отображается кнопка с названием учетной записи текущего пользователя. По нажатии на кнопку открывается меню со следующими пунктами:

- **Документация** ( ) — открывает руководство по эксплуатации системы.
- **Настройки профиля** ( ) — открывает страницу настроек профиля текущего пользователя.
- **Настройки сервера** ( ) — открывает окно настроек серверов для подключения к другим продуктам R-Vision.
- **Тема оформления** ( ) — переключает тему интерфейса (темная/светлая/синяя).
- **Выйти из системы** ( ) — прекращает текущий сеанс пользователя и осуществляет его выход из системы.

В верхнем правом углу расположено поле **Поиск** для глобального поиска по объектам системы в рамках текущего тенанта. Поиск охватывает следующие сущности:

- активные списки;
- базы данных;
- воркеры;
- дашборды;
- домены;
- запуски;
- коллекторы;
- конвейеры;
- конфигурации;
- модели событий;
- переменные;
- плейбуки;
- политики аудита источников;
- правила агрегации;
- правила корреляции;
- правила нормализации;
- резолюции;
- секреты;
- сканеры;
- скрипты;
- сформированные отчеты;

- схемы активных списков;
- схемы дашбордов;
- схемы доменов;
- схемы конфигураций;
- схемы шаблонов отчетов;
- таблицы обогащения;
- триггеры;
- уязвимости;
- хранилища событий;
- шаблоны отчетов.

В левой части окна расположено **главное меню** с основными разделами системы. В верхней части главного меню отображается название продукта, его текущая версия и меню выбора тенанта. В нижней части главного меню расположена кнопка со стрелкой для его сворачивания/разворачивания.

При нажатии на название раздела система откроет этот раздел и отобразит его содержимое в **рабочей области**, расположенной в центральной части окна.

В верхней части рабочей области расположена **панель инструментов**. Вид панели инструментов зависит от открытого раздела. На панели инструментов могут отображаться:

- Название раздела или элемента системы.
- Выпадающие списки, кнопки и поля для управления разделом.
- Информация о разделе.

Разделы могут отображать таблицу элементов. Набор возможностей для работы с таблицей зависит от открытого раздела.

В системе доступны следующие разделы:

- Группа разделов **Автоматизация**:
  - **Запуски** — раздел управления запусками скриптов.
  - **Резолюции** — раздел работы с запросами, поступившими на согласование в ходе работы плейбуков.
  - **Триггеры** — раздел управления условиями запуска скриптов.
- Группа разделов **Сканирование**:
  - **Задачи** — раздел управления задачами сканирования.
  - **Профили** — раздел управления профилями сканирования.
  - **Учетные записи** — раздел управления учетными данными для аутентификации на оборудовании в процессе его инвентаризации и поиска уязвимостей.
  - **Менеджер обновлений** — раздел загрузки и обновления базы уязвимостей.
- Группа разделов **Ресурсы**:

- **Сканеры** — раздел управления сканерами уязвимостей.
- **Домены** — раздел создания и настройки доменов системы.
- **Воркеры** — раздел управления воркерами.
- **Переменные** — раздел управления переменными, используемыми в плейбуках.
- **Конфигурации** — раздел управления конфигурациями ресурсов.
- **Активные списки** — раздел создания и настройки активных списков.
- **Коллекторы** — раздел создания и настройки коллекторов.
- **Модели событий** — раздел создания и настройки модели данных.
- **Базы данных событий** — раздел регистрации и настройки баз данных.
- **Хранилища событий** — раздел создания и настройки хранилищ событий.
- **Секреты** — раздел создания и настройки секретов для безопасного хранения чувствительных данных.

Если в настройках сервера системы настроена интеграция с R-Vision Endpoint, в списке также отображается раздел **Агенты**.

- Группа разделов **Агенты**:
  - **Агенты** — раздел управления агентами.
  - **Группы агентов** — раздел для работы с группами агентов.
  - **Шлюзы** — раздел для работы со шлюзами, с помощью которых агенты могут подключаться к кластерам.
- **Управление уязвимостями** — работа с уязвимостями, обнаруженными системой при сканировании оборудования.
- **Проверки соответствия** — работа с информацией, полученной при запуске задач сканирования с целью проверки соответствия системы установленным нормативным требованиям и стандартам безопасности.
- **Дашборды** — раздел аналитики по зарегистрированным событиям.
- **Поиск** — раздел для работы с событиями, зарегистрированными в системе, и записями активных списков.
- **Экспертиза** — раздел создания и настройки правил обработки событий безопасности, а также схем доменов, скриптов, плейбуков и схем конфигурации.
- **Отчеты** — раздел для работы с отчетами в системе.
- Группа разделов **Инструменты**:
  - **Аудит источников** — раздел для контроля состояния источников событий с помощью политик аудита источников.
  - **Мониторинг** — раздел наблюдения за состоянием кластера Kubernetes.
  - **Диспетчер запросов** — раздел для сбора и хранения данных об RQL-запросах.
  - **Политики управления доступом к событиям** — раздел настройки политик доступа пользователей к событиям в хранилищах на основе их ролей.

Настройки системы включают следующие разделы:

- Группа разделов **Управление доступом**:
  - **Группы пользователей** — раздел для создания и управления группами пользователей.
  - **Пользователи** — раздел для создания и управления пользователями.
  - **Роли** — раздел для создания и управления ролями пользователей.
  - **Тенанты** — раздел для управления тенантами.
  - **Политики входа и пароля** — раздел для управления аспектами безопасности пароля и правилами доступа к системе.
- Группа разделов **Пространства и сервисы**:
  - **Пространства** — раздел для управления подключениями к пространствам.
  - **Сервисы** — раздел для управления сервисами в пространствах.
- **LDAP-соединения** — раздел для управления LDAP-соединениями в системе.
- **Лицензия** — раздел для управления лицензией в системе.
- **Агенты** — раздел для настройки работы агентов.

Документация

Чтобы просмотреть руководство по эксплуатации из веб-интерфейса системы:

1. Откройте меню вашей учетной записи в правом верхнем углу экрана.
2. В открывшемся выпадающем меню выберите пункт **Документация** (). Система отобразит страницу руководства по эксплуатации системы в новой вкладке браузера.

Настройки профиля

Чтобы открыть настройки профиля:

1. Откройте выпадающее меню вашей учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Настройки профиля** (). Откроется окно настроек профиля.

Настройки профиля сгруппированы по следующим секциям:

- **Личные данные** — отображает основные сведения о пользователе: логин, ФИО, должность и описание.
- **Общие настройки интерфейса** — предлагает опции для выбора языка интерфейса системы и формата отображения дат.
- **Смена пароля** — предназначена для изменения текущего пароля пользователя.
- **Доступные разрешения** — отображает таблицу всех разрешений пользователя. Таблица содержит следующие столбцы:
  - **Разрешение** — название разрешения.
  - **Раздел** — раздел системы, который затрагивается разрешением. Путь до раздела

указывается через /.

- **Модуль** — функциональный модуль платформы, к которому относится разрешение.
- **Тенанты и организации** — список тенантов, в рамках которых выдано разрешение.

При работе с таблицей разрешений доступны следующие операции:

- Поиск разрешений по всем полям.
- Сортировка разрешений по полям **Разрешение** и **Модуль**.

Для учетных записей со статусом **Суперадминистратор** список разрешений не приводится, поскольку они обладают всеми доступными в системе разрешениями. Для таких пользователей в секции отображается только статус **Суперадминистратор**.

Настройки интерфейса

В системе доступны следующие настройки интерфейса:

- выбор языка;
- выбор формата даты;
- выбор часового пояса.

Выбор языка

Система поддерживает два языка интерфейса: **русский** и **английский**.

Чтобы изменить язык интерфейса:

1. Откройте выпадающее меню вашей учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Настройки профиля** (). Откроется окно настроек профиля.
3. В поле **Язык** секции **Общие настройки интерфейса** выберите из один из доступных языков: **Русский** или **Английский**.
4. Нажмите на кнопку **Сохранить**. Система автоматически сменит язык интерфейса на выбранный.

Выбор языка также доступен на странице авторизации системы.

Выбор формата даты

В системе преднастроены два формата дат и времени:

- `DD.MM.YYYY HH:MM:SS`, где:
  - `DD.MM.YYYY` — дата в формате "день.месяц.год".
  - `HH:MM:SS` — время в формате "час:минута:секунда".

При использовании формата `DD.MM.YYYY HH:MM:SS` время указывается

относительно часового пояса пользователя.

- `YYYY-MM-DDThh:mm:ss+/-hh:mm`, где:
  - `YYYY-MM-DD` — дата в формате "год-месяц-день".
  - `T` — начало обозначения времени.
  - `hh:mm:ss` — время в формате "час:минута:секунда".
  - `+hh:mm` или `-hh:mm` — разница между часовым поясом пользователя и часовым поясом UTC+0.

По умолчанию в системе используется формат `DD.MM.YYYY HH:MM:SS`.

Чтобы изменить формат даты:

1. Откройте выпадающее меню вашей учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Настройки профиля** (). Откроется окно настроек профиля.
3. В поле **Формат даты** секции **Общие настройки интерфейса** выберите из один из доступных форматов: **DD.MM.YYYY HH:MM:SS** или **YYYY-MM-DDThh:mm:ss+/-hh:mm**.
4. Нажмите на кнопку **Сохранить**. Система автоматически сменит формат отображения дат и времени на выбранный.

Выбор часового пояса

Время в системе определяется на основе часового пояса, выбранного пользователем.

Чтобы изменить часовой пояс:

1. Откройте выпадающее меню вашей учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Настройки профиля** (). Откроется окно настроек профиля.
3. В поле **Часовой пояс** секции **Общие настройки интерфейса** выберите необходимый часовой пояс.
4. Нажмите на кнопку **Сохранить**. Система автоматически пересчитает временные метки относительно выбранного часового пояса.

Смена пароля

Смена пароля недоступна пользователям из домена.

Чтобы изменить текущий пароль вашей учетной записи:

1. Откройте выпадающее меню вашей учетной записи в правом верхнем углу экрана.
2. Выберите пункт **Настройки профиля** (). Откроется окно настроек профиля.
3. В секции **Смена пароля** укажите текущий пароль, затем введите новый пароль и повторите его для подтверждения.
4. Нажмите на кнопку **Изменить пароль**. Если новый пароль удовлетворяет заданным требованиям, система сохранит его и отобразит соответствующее уведомление.

Работа с таблицами

Списки объектов системы, как правило, организованы в виде таблиц, сгруппированных по различным системным разделам.

Система предлагает следующие возможности для работы с таблицей:

- пагинация;
- настройка ширины столбцов;
- просмотр элементов;
- выделение элементов;
- сортировка элементов;
- фильтрация элементов;
- поиск элементов;
- обновление таблицы;
- настройка отображения таблицы.

Набор доступных возможностей зависит от раздела системы.

Пагинация таблицы

Пагинация позволяет оптимизировать работу с таблицей путем структурирования и группировки ее элементов по страницам.

Панель пагинации располагается в нижней части таблицы и содержит следующие компоненты для управления страницами:

- Кнопки ‹ и › позволяют перейти к предыдущей и следующей странице соответственно.
- Кнопки « и » позволяют перейти к первой и последней странице соответственно.
- Поле ввода, отображающее номер открытой страницы, позволяет ввести номер нужной страницы для быстрого перехода к ней.
- Выпадающий список позволяет настроить количество отображаемых на странице элементов: 50, 100, 200 или 500. По умолчанию установлен лимит в 50 элементов.

При изменении количества отображаемых элементов изменяется и общее количество страниц в таблице.

Выбранное количество элементов сохраняется до конца сеанса пользователя.

Изменение ширины столбцов таблицы

По умолчанию все столбцы в таблице имеют одинаковую ширину. Стандартная ширина столбцов не всегда достаточна для корректного отображения их содержимого. Для лучшего отображения данных ширину столбцов можно изменить.

Чтобы изменить ширину столбца таблицы:

1. Наведите курсор на правую границу заголовка нужного столбца. Курсор примет вид двунаправленной стрелки.
2. Перетащите правую границу заголовка в нужном направлении.

Двойным нажатием по правой границе заголовка можно автоматически подогнать ширину столбца под его содержимое.

Просмотр элементов таблицы

Просмотр элемента таблицы позволяет получить детальную информацию об этом элементе: например, его название, идентификатор, тенант, даты создания и изменения.

Просмотр сведений об элементе осуществляется с помощью его карточки. Чтобы открыть карточку, нажмите на нужный элемент в таблице. Способ отображения карточки и ее содержание зависят от раздела системы.

Выделение элементов таблицы

Выделение позволяет выполнять массовые операции над элементами таблицы: например, экспортировать или удалять их.

Есть два способа выделения:

- Чтобы выделить один или несколько элементов, установите флажки напротив этих элементов в левом столбце таблицы.
- Чтобы выделить все элементы на странице, установите флажок в заголовке первого столбца таблицы. При этом будут выделены только элементы текущей страницы. Чтобы выделить большее число элементов, перейдите на следующую страницу и повторите действие.

Выделение сохраняется при переходе между страницами и изменении количества отображаемых элементов.

Общее количество выделенных элементов и доступные операции над выделенными элементами отображаются на панели инструментов.

Сортировка элементов таблицы

Сортировка позволяет упорядочить элементы таблицы по какому-либо столбцу в выбранном направлении.

Чтобы отсортировать таблицу, нажмите на кнопку действий (⋮) в заголовке нужного столбца и выберите направление сортировки:

- **По умолчанию** ( ) — сортировка отсутствует.
- **Возрастание** ( ).
- **Убывание** ( ).

Применить сортировку также можно с помощью аналогичных кнопок в заголовке столбца: последовательное нажатие кнопки меняет направление сортировки.

Фильтрация элементов таблицы

Фильтрация позволяет отобразить в таблице только те элементы, которые удовлетворяют заданным условиям.

В системе доступны следующие способы фильтрации:

- через боковую панель;
- через выпадающий список.

Способ фильтрации зависит от раздела системы.

Фильтрация через боковую панель

Чтобы отфильтровать таблицу через боковую панель:

1. Нажмите на кнопку **Фильтр** (), расположенную над таблицей. В правой части рабочей области откроется панель настройки фильтрации.
2. Задайте настройки фильтрации на панели. В таблице отобразятся только элементы, удовлетворяющие заданным условиям.

Чтобы сбросить условие фильтрации, очистите соответствующее поле с помощью кнопки ✕. Чтобы сбросить сразу все условия, нажмите на кнопку **Сбросить** в нижней части панели.

Фильтрация через выпадающий список

Чтобы отфильтровать таблицу через выпадающий список, выберите нужный пункт в списке, расположенном над таблицей. В таблице отобразятся только элементы с выбранным критерием.

Чтобы сбросить условие фильтрации, снимите выбор в выпадающем списке с помощью кнопки ✕.

Поиск элементов таблицы

Поиск позволяет быстро находить в таблице нужные элементы по содержимому их полей.

Чтобы выполнить поиск, введите запрос в строку поиска, расположенную над таблицей, и нажмите клавишу ENTER. В таблице отобразятся элементы, в соответствующих полях которых содержится введенный запрос.

Особенности поиска:

- Поиск осуществляется по вхождению указанных символов в полях **ID** и **Название**.
- При поиске учитывается строгая последовательность введенных символов, вхождение которой проверяется в любом месте строки.
- При поиске не учитывается регистр символов.

Обновление таблицы

Во время работы с таблицей перечень ее элементов может обновиться. При обновлении таблицы

изменения не отображаются автоматически в интерфейсе программы.

Чтобы отобразить изменения в таблице, ее требуется обновить. Вы можете обновить таблицу вручную или настроить автоматическое обновление.

Обновление таблицы вручную

Чтобы обновить таблицу вручную, нажмите на кнопку **Обновить** (🔄) в нижней части таблицы. Перечень элементов таблицы будет обновлен.

Автоматическое обновление таблицы

Чтобы настроить регулярное автоматическое обновление таблицы:

1. Нажмите на кнопку **Автообновление** в нижней части таблицы. Отобразится панель настроек автоматического обновления.
2. Задайте настройки автоматического обновления:

   a. Укажите количество единиц времени, по истечении которых будет происходить обновление. Вы можете сделать это вручную или с помощью кнопок — и +.

   b. Выберите из выпадающего списка единицы времени: секунды, минуты, часы.

   c. Переведите переключатель **Включить автообновление** в активное положение.

   Перечень элементов таблицы будет автоматически обновляться по истечении указанного периода времени.

Настройка отображения таблицы

Настройки отображения таблицы сохраняются индивидуально для каждого пользователя системы.

При необходимости вы можете задать настройки отображения таблицы для повышения удобства работы с ней.

Чтобы настроить внешний вид таблицы:

1. Нажмите на кнопку ⊞ в нижней части таблицы. Система отобразит в правой части экрана панель настроек таблицы.
2. Настройте отображение таблицы:

   a. Выберите высоту строки: стандартную или компактную.

   b. Установите флажки рядом с названиями полей, которые должны отображаться в таблице.

   c. Разместите поля в желаемом порядке, перетаскивая их с помощью мыши, чтобы определить порядок их отображения в таблице. Для этого наведите курсор на иконку ═ слева от флажка и перетащите поле в нужную позицию.

   Чтобы применить к таблице настройки отображения по умолчанию, нажмите на кнопку **Восстановить по умолчанию**.

Клавиши и комбинации клавиш

Клавиши и их комбинации для работы в интерфейсе системы.

Общие

Функция	Клавиши и их комбинации
Перейти на предыдущую строку и выделить ее	SHIFT+↑
Перейти на следующую строку и выделить ее	SHIFT+↓
Перейти на первую строку в списке (на текущей странице) и выделить ее	CTRL+SHIFT+↑
Перейти на последнюю строку в списке (на текущей странице) и выделить ее	CTRL+SHIFT+↓
Перейти на следующую страницу	SHIFT+→
Перейти на предыдущую страницу	SHIFT+←
Перейти на последнюю страницу	CTRL+SHIFT+→
Перейти на первую страницу	CTRL+SHIFT+←
Перейти к следующему полю формы	TAB
Перейти к предыдущему полю формы	SHIFT+TAB
Закрыть с отменой (отправки формы)	ESC
Закрыть с подтверждением (отправки формы)	ENTER
Развернуть карточку выбранного элемента	ENTER
Свернуть карточку выбранного элемента	ESC

Поиск по событиям

Функция	Клавиши и их комбинации
Перейти в строку поиска	SHIFT+R
Запустить поиск	SHIFT+ENTER
Остановить текущий поиск и запустить новый (перезапустить)	CTRL+SHIFT+ENTER
Открыть форму сохранения поискового запроса как шаблона	CTRL+S
Открыть форму добавления фильтра	SHIFT+F

Конфигурация конвейера

Функция	Клавиши и их комбинации
Удалить выделенный элемент конвейера или связь	DELETE (в Windows также BACKSPACE)

Редактор кода

Функция	Клавиши и их комбинации
Сместить табуляцию на шаг вправо	TAB
Сместить табуляцию на шаг влево	SHIFT+TAB
Вызвать подсказки кода Чтобы вставить выбранную подсказку, нажмите TAB или ENTER.	CTRL+SPACE
Искать по коду	CTRL+F
Включить или выключить поиск с учетом регистра	ALT+C
Включить или выключить поиск слова целиком	ALT+W
Включить или выключить поиск с регулярными выражениями	ALT+R
Выделить текущее слово.	CTRL+D Чтобы выделить следующее идентичное значение и добавить курсор после него, удерживая CTRL, нажмите D.
Добавить курсор на следующие строки	CTRL+ALT+↓
Добавить курсор на предыдущие строки	CTRL+ALT+↑
Выделить всю строку	CTRL+L
Выделить следующую строку	SHIFT+↓
Выделить предыдущую строку	SHIFT+↑
Выделить весь код	CTRL+A
Копировать выделенное в буфер обмена	CTRL+C
Вырезать выделенное	CTRL+X
Вставить из буфера обмена	CTRL+V
Переместить текущую строку вниз	ALT+↓
Переместить текущую строку вверх	ALT+↑
Дублировать текущую строку и переместить курсор вниз	SHIFT+ALT+↓
Дублировать текущую строку и оставить курсор на текущей строке	SHIFT+ALT+↑
Отменить последнее действие	CTRL+Z
Повторить отмененное действие	CTRL+Y

Начало работы с системой

Для начала работы необходимо настроить обработку событий ИБ в системе. Настройка включает следующие этапы:

1. Регистрация баз данных событий.
2. Создание моделей событий.
3. Создание хранилищ событий с указанием используемой базы данных и модели события.
4. Настройка элементов экспертизы.
5. Настройка активных списков на основе схем.
6. Создание секретов.
7. Настройка коллекторов.

В следующих разделах приведены примеры настройки конвейеров обработки событий:

- Обработка событий на одном конвейере
- Обработка событий на нескольких конвейерах
- Обработка событий на нескольких коллекторах
- Обработка событий с нескольких IP-адресов
- Сбор событий из вынесенного коллектора

Обработка событий на одном конвейере

Данное руководство описывает процесс создания коллектора и конвейера для сбора и обработки событий.

Об обработке событий в коллекторе

Для работы с событиями нужно создать **коллектор** — компонент, который обеспечивает сбор событий из источников, их обработку и дальнейшую отправку на хранение и/или во внешние системы. События поступают в коллектор через установленные на нем конвейеры, посредством которых они могут быть обработаны.

Конвейер — это упорядоченная последовательность взаимосвязанных этапов обработки, представленных такими элементами, как коррелятор, нормализатор, агрегатор, VRL-трансформация и фильтр. Эти элементы конфигурируются на основе правил и условий, сформулированных на VRL — языке описания трансформаций и корреляций.

С помощью маршрутизатора и связей между элементами конвейера можно управлять порядком обработки событий. После обработки события можно отправлять в связанные с конвейером хранилища, а также во внешние системы. Хранилище находится в базе данных в виде таблицы, формат которой определяется моделью событий хранилища.

В данной статье рассмотрен процесс настройки коллектора, в котором все события собираются и обрабатываются в рамках единственного конвейера.

В полном цикле настройки системы для приема и обработки событий можно выделить следующие этапы:

1. Создание модели и хранилища событий
    a. Создание модели события
    b. Регистрация базы данных событий
    c. Создание хранилища событий
2. Создание коллектора и конвейера
    a. Создание коллектора
    b. Создание конвейера
3. Добавление элементов конвейера
    a. Добавление точки входа
    b. Добавление VRL-трансформации
    c. Добавление маршрутизатора
    d. Добавление фильтра
    e. Настройка агрегации событий
    f. Настройка нормализации событий
    g. Настройка корреляции событий
    h. Добавление конечной точки
    i. Объединение элементов конвейера
4. Включение коллектора и установка конфигурации конвейера

Следуйте шагам настройки в указанной последовательности. Добавить элементы конвейера можно в любом порядке. Однако соответствующие элементы экспертизы нужно создавать перед добавлением элементов конвейера.

Примеры конкретных конфигураций приведены после общего описания действий.

Создание модели и хранилища событий

Создание модели события

При создании хранилища нужно обязательно указать используемую в нем **модель события**. Модель определяет формат хранилища событий: имена, типы полей и данных в полях.

Чтобы создать модель события:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания модели события.
3. Введите название модели события.
4. Введите описание модели события.

5. В нижней части окна находится раздел **Схема**, содержащий список полей, которые будут использоваться в этой модели события.

   По умолчанию система отображает набор **служебных** полей.

   Список можно расширить, добавив вручную **регулярные** поля. Для этого:

   a. Нажмите на кнопку **Добавить поле**. Система отобразит окно добавления поля.

   b. Введите уникальный ключ идентификации нового поля.

   Ключ может состоять только из латинских букв, цифр и подчеркиваний. При этом он не может начинаться с цифры.

   c. При необходимости установите флажок **Поле с изменяемой семантикой**, чтобы использовать значения из другого поля при построении семантики названия нового поля.

   Выберите поле из выпадающего списка **Поле, хранящее название**, чтобы определить, какое поле будет использовать система для формирования его названия. Можно использовать только следующие типы полей: **String**, **LCString**, **Enum**.

   d. Введите описание нового поля.

   e. Выберите тип данных нового поля. В зависимости от выбранного типа данных необходимо указать его параметры:

   - **Array** — выберите тип элемента **Array** из списка.
   - **Enum** — укажите значения Enum, разделяя их запятой без пробелов.
   - **KeyValue** — выберите тип элемента **KeyValue** из списка.
   - **UInt** — выберите разрядность из списка.

   f. При необходимости включите переключатель **Может быть null**, чтобы поле могло хранить значение типа `null`. Если переключатель выключен, а поле не заполнено, событие в хранилище не будет создано.

   В полях типов **Array**, **KeyValue**, **LCString** этот переключатель не используется.

   g. При необходимости включите переключатель **Отображать поле**, чтобы поле отображалось в списке результатов поиска в разделе **Поиск**.

   Эту настройку можно изменить при просмотре модели события или при настройке внешнего вида списка событий.

   h. Нажмите на кнопку **Добавить**.

   Повторите действия, чтобы добавить несколько регулярных полей.

   По умолчанию, универсальная модель содержит 150 служебных полей. Пользователь может добавить до 20 регулярных полей.

При необходимости вы можете изменить или удалить добавленные регулярные поля:

- Чтобы изменить поле, нажмите на кнопку в строке поля. Отобразится окно изменения поля, в котором внесите требуемые изменения и нажмите на кнопку **Сохранить**. Измененные данные поля будут сохранены.
- Чтобы удалить поле, нажмите на кнопку в строке поля. Поле будет удалено из списка полей модели события.

Служебные поля в модели данных создаются и заполняются автоматически и недоступны для изменения или удаления.

6. Нажмите на кнопку **Создать**. Система создаст модель событий и отобразит соответствующее уведомление. Новая модель появится в списке раздела **Ресурсы** → **Модели событий**.

Регистрация базы данных событий

Базы данных (БД) событий используются для хранения событий в системе. Единицей хранения событий в БД является хранилище событий. На уровне БД хранилище представляет собой таблицу. Столбцы и типы данных хранилища задаются моделью события.

Чтобы зарегистрировать БД событий:

1. Перейдите в раздел **Ресурсы** → **Базы данных событий**. Система отобразит сведения об имеющихся БД событий.
2. Нажмите на кнопку **Зарегистрировать** ( ). Отобразится окно регистрации БД событий.
3. Введите название БД событий.
4. Введите идентификатор БД ClickHouse в NATS Service Discovery.

Чтобы получить идентификатор БД ClickHouse в NATS Service Discovery, обратитесь к администратору системы или воспользуйтесь разделом Получение идентификатора сервиса в NATS Service Discovery.

5. При необходимости введите описание БД событий.
6. Нажмите на кнопку **Зарегистрировать**. Система создаст БД событий и отобразит соответствующее уведомление. Новая база отобразится в списке раздела **Ресурсы** → **Базы данных событий**.

Создание хранилища событий

Хранилище событий используется для сохранения событий, поступивших в систему и обработанных коллектором. Оно представляет собой таблицу в БД событий. Формат хранилища определяется его моделью событий.

Чтобы создать хранилище событий:

1. Перейдите в раздел **Ресурсы** → **Хранилища событий**. Система отобразит сведения об имеющихся хранилищах событий.
2. Нажмите на кнопку **Создать** ( ). Отобразится окно создания хранилища событий.

3. Выберите из выпадающего списка БД событий, в которую требуется поместить хранилище. Система отобразит в нижней части поля для настройки томов хранения этой БД.

В системных базах данных создание хранилищ событий не поддерживается.

4. Выберите из выпадающего списка модель события, которая будет использована для создаваемого хранилища.
5. Введите название хранилища событий.
6. При необходимости введите описание хранилища событий.
7. Задайте настройки хранения данных:

   a. Выберите том хранения данных:

      - **hot_volume** — том для быстрого доступа к часто используемым данным (горячее хранение).
      - **cold_volume** — том для хранения редко используемых данных (холодное хранение).

   b. Укажите срок хранения данных в днях.

Если поле для срока хранения данных оставить пустым, то срок хранения данных в этом хранилище будет неограниченным.

   c. Выберите из выпадающего списка действие над данными в томе по окончании срока хранения:

      - **Переместить** — данные будут перемещены в другой том. Введите в появившейся строке настройки нового тома, в который требуется переместить данные.
      - **Удалить** — данные в томе будут удалены.

   При необходимости можно задать настройки хранения данных на основе настроек по умолчанию в выбранной БД. Для этого нажмите на кнопку **Применить настройки хранения по умолчанию БД**. Если настройки по умолчанию не заданы, система отобразит соответствующее сообщение.

8. Нажмите на кнопку **Создать**. Система создаст хранилище событий и отобразит соответствующее уведомление. Новое хранилище появится в списке раздела **Ресурсы → Хранилища событий**.

Создание коллектора и конвейера

Создание коллектора

Перед тем как система начнет принимать события от источника данных, необходимо создать соответствующий коллектор.

Чтобы создать коллектор:

1. Перейдите в раздел **Ресурсы → Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания коллектора.
3. Введите название коллектора.

4. Выберите подключение к пространству, в котором нужно создать коллектор. По умолчанию коллектор создается в центральном пространстве.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок + и —.

- При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов в каждом узле кластера Kubernetes.
- Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.
- Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

6. При необходимости введите описание коллектора.
7. Для работы в режиме распределенной корреляции переведите переключатель **Распределенный коррелятор** в активное положение.

Изменение этой настройки потребует перезапуска коллектора.

Если система работает на базе кластера Kubernetes с одним узлом, включение режима распределенной корреляции недоступно.

8. Укажите количество ядер процессора (**CPU**) и объем оперативной памяти (**RAM**), выделяемый для работы распределенного коррелятора. Это можно сделать непосредственно в поле или с помощью кнопок + и —.

Операции полей `join` и `group` декларативного правила вычисляются на коррелляторе. Остальные поля, в том числе `alias` и `filter`, вычисляются на предпроцессорах. В составе распределенного коррелятора может быть только один коррелятор, но несколько предпроцессоров.

a. Укажите ресурсы для коррелятора:
   i. Выберите узел для размещения коррелятора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.
   ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы коррелятора. Количество доступных в узле ресурсов отображается под полями ввода.
b. Укажите ресурсы для предпроцессора:
   i. Выберите узел для размещения предпроцессора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.
   ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы предпроцессора. Количество доступных в узле ресурсов отображается под полями ввода.

9. Нажмите на кнопку **Добавить предпроцессор**, если необходимо добавить дополнительный предпроцессор.
10. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:
    - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
    - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
    - **Info** — записывается общая информация о работе и состоянии коллектора.
    - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
    - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.
11. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее уведомление. Новый коллектор появится в списке раздела **Ресурсы** → **Коллекторы**.

Создание конвейера

Чтобы добавить конвейер:

1. Находясь в разделе **Ресурсы** → **Коллекторы**, нажмите на строку коллектора, созданного ранее. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
2. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
3. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
4. Введите название конвейера.
5. При необходимости введите описание конвейера.

Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

6. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке на вкладке **Конвейеры** в карточке коллектора.

Добавление элементов конвейера

При настройке конвейера необходимо создать его компоненты, сконфигурировать их, упорядочить и объединить в последовательность, соответствующую логике обработки данных.

Чтобы настроить конвейер, требуется открыть его конфигурацию в режиме черновика. Если при создании конвейера флажок **Открыть конфигурацию конвейера** не был установлен, то выполните одно из действий:

- Нажмите на стрелку в строке конвейера (⌄). Откроется карточка конвейера. Нажмите на кнопку

Конфигурация конвейера в нижней части карточки.

- Нажмите на кнопку действий (⋮) в строке конвейера и выберите опцию **Конфигурация конвейера**.

В открытом окне конфигурации конвейера убедитесь, что в выпадающем списке **Версия** в правом верхнем углу окна выбран пункт **Черновик**.

Далее рассмотрим сценарий настройки конфигурации конвейера для обработки исходных событий в формате JSON.

▼ *Пример исходного события*

```
{
  "event": "{\"host\":\"198.51.100.191\",\"user-
identifier\":\"JohnDoe\",\"datetime\":\"31/Jul/2023:06:55:49\",\"method\":\"GET\",\"
request\":\"/observability/metrics/production\",\"protocol\":\"HTTP/2.0\",\"status\"
:\"404\",\"bytes\":22427,\"referer\":\"https://example.com\"}"
}
```

Добавление точки входа

Точка входа поставляет в систему поток сырых событий. Выбор формата точки входа зависит от формата данных, получаемых из источника события. Точка входа имеет один выход и не имеет входов.

Для отладки конвейера вы можете использовать точку входа Demo Logs, которая генерирует случайные события.

Чтобы добавить точку входа на конвейер:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Точка входа**. Отобразится окно добавления точки входа.
2. При необходимости выберите шаблон для автоматического заполнения полей точки входа. По умолчанию выбран вариант **Без шаблона**.
3. Заполните поля (набор и содержание полей могут отличаться в зависимости от типа точки):
    - Название точки входа.
    - Тип точки входа.
    - Политика аудита источников.

Точка входа может быть привязана только к одной политике аудита источников, но к одной политике может быть привязано несколько точек входа.

    - Формат точки входа.
    - Данные для подключения.
    - Настройки шифрования соединения (опционально).

- Количество событий (опционально).
- Интервал в секундах (опционально).

4. Установите флажок **Сохранить как шаблон**, если настройку точки входа необходимо сохранить в качестве шаблона для дальнейшего использования.
5. Нажмите **Добавить**. Новая точка входа отобразится на диаграмме конфигурации.

Пример:

Создадим источник, который будет поставлять логи HTTP-сервера в формате JSON.

Для этого выберем тип точки входа Demo Logs и формат генерируемых событий **json**. Укажем в названии точки входа "JSON".

Добавление VRL-трансформации

VRL-трансформация служит для обработки и модификации событий с применением языка VRL. Этот элемент конвейера принимает на вход события, проводит их преобразование и на выходе предоставляет уже модифицированные события.

Чтобы добавить VRL-трансформацию на конвейер:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **VRL-трансформация**. Отобразится окно добавления VRL-трансформации.
2. При необходимости выберите шаблон для автоматического заполнения полей VRL-трансформации. По умолчанию выбран вариант **Без шаблона**.
3. Введите название VRL-трансформации.
4. Введите правило, по которому будет преобразовываться событие, на языке VRL.

В редакторе кода доступны горячие клавиши.

В VRL-программе для доступа к данным события используется символ точки (`.`), а к метаданным события — символ процента (`%`). В корреляционных событиях метаданные включают данные исходных событий.

5. При необходимости из выпадающего списка **Обработка ошибок** выберите шину коллектора, в которую будет отправлено событие в случае возникновения ошибок обработки. По умолчанию выбран вариант **События с ошибкой**.
6. Установите флажок **Сохранить как шаблон**, если настройку VRL-трансформации необходимо сохранить в качестве шаблона для дальнейшего использования.
7. Нажмите на кнопку **Добавить**. Новая VRL-трансформация отобразится на диаграмме конфигурации.

Пример:

Добавим условие на языке VRL, согласно которому исходное событие будет обогащаться дополнительным полем `responseStatus`. В нем будет устанавливаться значение "Bad request", если в поле `status` приходящего на VRL-трансформация события содержится значение 401 или 404.

Для этого создадим VRL-трансформацию с названием "AssignResponseStatus" и пропишем в свойствах следующее условие:

▼ *Пример условия обогащения событий, язык VRL*

```
if .status == "404" || .status == "401" {
.responseStatus = "Bad request"
}
```

▼ *Интерпретация*

`if .status == "404" || .status == "401"`:

- Условное выражение проверяет, равно ли значение поля `status` события строке "404" или "401".

`.responseStatus = "Bad request"`:

- Если условие выше выполняется, то поле `responseStatus` текущего события устанавливается в значение "Bad request".

Добавление маршрутизатора

Маршрутизатор направляет события на разные этапы обработки в зависимости от их содержимого или других условий. Маршрутизатор имеет один вход и несколько выходов. На входе каждое событие проверяется по условиям и появляется на тех выходах, где условие выполняется.

Чтобы добавить маршрутизатор на конвейер:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Маршрутизатор**. Отобразится окно добавления маршрутизатора.
2. Введите название маршрутизатора.

   Для маршрутизатора недопустимо использование названий "_unmatched" и "_default".

3. Настройте список маршрутов маршрутизатора.
   a. Нажмите кнопку **Добавить маршрут**.
   b. Введите название маршрута.
   c. Задайте правило обработки событий на языке VRL.
   d. Нажмите на кнопку **Добавить**. Маршрут будет добавлен в маршрутизатор.
4. При необходимости установите флажок **Добавить маршрут по умолчанию**. В этом случае к маршрутизатору будет добавлен маршрут **По умолчанию**, на выход которого будут попадать те события, которые не удовлетворяют условиям остальных маршрутов.

   При неустановленном флажке события, не удовлетворяющие условиям ни одного из маршрутов, будут удаляться.

5. Нажмите на кнопку **Добавить**. Новый маршрутизатор отобразится на диаграмме конфигурации.

Пример:

Добавим проверку статуса события, после которой:

- События со статусами 400, 401, 403 и 404 отправляются на дальнейшую обработку.
- События со статусами 500, 502 и 504 отправляются на хранение.

Для этого создадим маршрутизатор с названием "StatusCheckRouter", добавим два маршрута, а также маршрут по умолчанию.

Первому маршруту дадим название "ClientErrorRoute" и следующее условие:

Пример 21. Пример условия фильтрации событий для первого маршрута, схема RObject

```
.event.status == "400" || .event.status == "401" || .event.status == "403" ||
.event.status == "404"
```

Второму маршруту дадим название "ServerErrorRoute" и следующее условие:

Пример 22. Пример условия фильтрации событий для второго маршрута, схема RObject

```
.event.status == "500" || .event.status == "502" || .event.status == "504"
```

Добавление фильтра

Фильтр предназначен для сокращения объема обрабатываемых событий путем их отсеивания на основе заданных условий. Условия обычно формулируются через имена и значения полей, и для их определения используются математические и логические операторы. Например, условие `externalId == "4648" && categoryOutcome == "Success"` позволяет пропустить через фильтр только те события, которые удовлетворяют этим критериям. У фильтра есть один вход и один выход. На вход фильтра подается событие, и если оно соответствует заданным условиям, то проходит на выход. В противном случае событие отсеивается и не участвует в дальнейшей обработке.

Чтобы добавить фильтр на конвейер:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Фильтр**. Отобразится окно добавления фильтра.
2. При необходимости выберите шаблон для автоматического заполнения полей фильтра. По умолчанию выбран вариант **Без шаблона**.
3. Введите название фильтра.
4. Задайте условие фильтрации входящих событий на языке VRL.

   Условия обычно формулируются через имена полей и их значения, и для их определения используются математические и логические операторы.

В редакторе кода доступны горячие клавиши.

В VRL-программе для доступа к данным события используется символ точки (`.`),

а к метаданным события — символ процента (%). В корреляционных событиях метаданные включают данные исходных событий.

5. Установите флажок **Сохранить как шаблон**, если настройку фильтра необходимо сохранить в качестве шаблона для дальнейшего использования.
6. Нажмите на кнопку **Добавить**. Новый фильтр отобразится на диаграмме конфигурации.

Пример:

Добавим фильтр, который пропускает только события с хоста 198.51.100.191.

Для этого создадим фильтр с названием "FilterByHost", и в поле **Условие** пропишем следующее выражение:

Пример 23. Пример условия фильтрации событий, схема RObject

```
.event.host == "198.51.100.191"
```

Настройка агрегации событий

Чтобы события, проходящие через конвейер обработки событий, агрегировались, необходимо добавить агрегатор.

Чтобы добавить агрегатор, необходимо создать и установить на нем правило агрегации, согласно которому будут объединяться поступающие события.

Создание правила агрегации

Правило агрегации в системе является структурированным объектом данных в формате RObject (.ro), описывающим процесс объединения и анализа групп событий безопасности. Основная функция правила агрегации — сбор и объединение данных из различных событий в рамках заданной логики. Каждое правило агрегации включает в себя критерии для группировки событий, условия начала и завершения агрегации, а также стратегии агрегации для каждого поля. Агрегация инициируется в соответствии с условиями, определенными в правилах, и преобразует входящий поток событий в агрегированные данные.

Чтобы добавить правило агрегации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило агрегации**. Система отобразит окно создания правила агрегации.

▼ *Пример правила агрегации, схема RObject*

```
# Уникальный идентификатор правила.
id: example/network_activity_rule

# Название правила.
name: NetworkActivityAggregationRule
```

```
# Тип элемента экспертизы.
type: aggregation_rule

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Агрегация событий сетевой активности для анализа трафика

# Имя и контактные данные автора правила.
author: John Doe <johndoe@example.com>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_1
    - EventID_2

# Теги, связанные с правилом.
tags:
  - network
  - traffic_analysis

# Отбрасывает события, которые не соответствуют критериям.
filter: !vrl |
  # Фильтрует события, не относящиеся к категории "Network" или имеющие низкий
уровень серьезности.
  .category == "Network" &&
  to_int!(.severity) > 3

# Группировка событий по источнику и назначению.
group_by:
  - src_ip
  - dest_ip

# Условие открытия агрегационного окна.
start_when: !vrl |
  # Начинаем агрегацию при обнаружении события начала соединения.
  .event_type == "connection_start"

# Максимальное количество событий.
max_events: 5

# Время ожидания последнего события (миллисекунды).
expire_after_ms: 60000

# Интервал проверки окон агрегации (миллисекунды).
flush_period_ms: 2000
```

```
# Стратегии агрегации значений для каждого указанного поля.
merge_strategies:
  bytes_sent: sum
  bytes_received: sum
  severity: max

# Тесты для проверки работы правила.
tests:
 - name: Network Activity Test
   events:
   - {"category":"Network", "severity":5, "event_type":"connection_start",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":1500,
"bytes_received":2000}
   - {"category":"Network", "severity":4, "event_type":"data_transfer",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":3000,
"bytes_received":4000}
   - {"category":"Network", "severity":5, "event_type":"connection_end",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":500,
"bytes_received":600}
   assertion: !vrl |
     # Проверяем, что агрегированные значения соответствуют ожиданиям.
     assert_eq!(.[0], {
       "category": "Network",
       "event_type": "connection_start",
       "src_ip": "192.0.2.10",
       "dest_ip": "192.0.2.20",
       "bytes_sent": 5000,
       "bytes_received": 6600,
       "severity": 5
     })
```

▼ *Интерпретация*

Правило агрегации "NetworkActivityAggregationRule" служит для агрегации сетевых событий, соответствующих передаче данных в рамках одного сетевого соединения.

Метаданные правила:

- `id`: уникальный идентификатор правила в системе.
- `name`: название правила.
- `type`: тип `aggregation_rule` указывает на то, что это правило агрегации.
- `version`: версия правила.
- `description`: описывает цель правила — "Агрегация событий сетевой активности для анализа трафика".
- `author`: создатель правила.
- `tags`: теги `network` и `traffic_analysis` классифицируют правило по его функциональности и области применения.

Фильтрация событий (filter):

- `filter`: фильтрует события по категории "Network" и уровню серьезности более 3. Это сужает область применения правила только к событиям, которые удовлетворяют условиям фильтрации.

Группировка событий (group_by):

- `group_by`: группирует события по полям `src_ip` и `dest_ip`. Это позволяет анализировать взаимодействия между конкретными источниками и назначениями в сети.

Условие начала агрегации (starts_when):

- `starts_when`: определяет начало агрегации при появлении событий типа "connection_start", что является признаком начала нового сетевого соединения.

Параметры агрегационного окна (max_events, expire_after_ms, flush_period_ms):

- `max_events`: ограничивает количество событий в агрегации до 5. После достижения этого количества, агрегационное окно закрывается и генерируется агрегированное событие.
- `expire_after_ms`: устанавливает максимальный период ожидания последнего события в агрегационном окне в 60000 миллисекунд (1 минута).
- `flush_period_ms`: задает частоту проверки и закрытия агрегационных окон в 2000 миллисекунд (2 секунды).

Стратегии агрегации (merge_strategies):

- `merge_strategies`: определяет, как будут объединяться поля событий. Для "bytes_sent" и "bytes_received" применяется стратегия суммирования (`sum`).

Тестирование правила (tests):

- `name`: "Network Activity Test" — название теста.
- `events`: описывает ряд тестовых событий, имитирующих реальный сетевой трафик, для проверки эффективности правила.
- `assertion`: проверяет, что правило корректно агрегирует события и вычисляет суммы переданных и полученных байтов. Это подтверждает, что правило работает как предполагалось и агрегирует данные согласно заданным критериям.

▼ *Обработка событий правилом агрегации*

Проанализируем работу правила агрегации "NetworkActivityAggregationRule", фокусируясь на процессе обработки событий.

Представим, что на агрегатор поступает поток сетевых событий, которые соответствуют критериям, заданным в фильтре `filter` правила. Эти события содержат данные о типе сетевого соединения, IP-адресах источника и назначения, а также объеме переданных и полученных данных.

```
{
  "category": "Network",
```

```
  "severity": 5,
  "event_type": "connection_start",
  "src_ip": "192.168.1.10",
  "dest_ip": "192.168.1.20",
  "bytes_sent": 1500,
  "bytes_received": 2000
}
```

По мере поступления этих событий, они фильтруются правилом "NetworkActivityAggregationRule". Правило агрегирует события, группируя их по IP-адресам источника и назначения (`group_by`). Таким образом, события, исходящие от одной пары IP-адресов, объединяются в одно агрегированное событие.

Агрегация инициируется, когда обнаруживается событие типа `connection_start` (`starts_when`). Правило продолжает агрегировать события до тех пор, пока не будет достигнуто максимальное количество событий (`max_events`) или не истечет время ожидания (`expire_after_ms`).

Агрегированное событие включает в себя сумму значений `bytes_sent` и `bytes_received` для всех событий в группе (`merge_strategies`).

```
{
  "event_type": "aggregated_network_activity",
  "src_ip": "192.168.1.10",
  "dest_ip": "192.168.1.20",
  "total_bytes_sent": 5000,
  "total_bytes_received": 6600
}
```

Данное агрегированное событие отражает общую активность между двумя конкретными IP-адресами, позволяя идентифицировать аномалии или необычную активность в сети.

3. Заполните поля правила агрегации, чтобы определить логику его работы, и нажмите на кнопку **Опубликовать версию**. Система создаст правило и отобразит соответствующее уведомление. Новое правило агрегации отобразится в таблице раздела **Экспертиза**.
4. Чтобы созданное правило стало доступным на конвейерах событий, включите его:
   a. Нажмите на строку с созданным правилом агрегации. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
   b. Убедитесь, что в карточке открыта вкладка **Информация**.
   c. Переведите переключатель состояния в верхней части карточки в активное положение и подтвердите включение правила. Система обновит информацию о статусе правила агрегации.

Добавление агрегатора

Агрегатор представляет собой компонент конвейера, который использует правила агрегации для объединения и анализа групп событий безопасности.

Порядок применения правил агрегации определяется последовательностью их добавления в агрегатор. Это позволяет управлять процессом агрегации и определять, какие события будут объединены вместе.

Агрегатор на входе принимает поток событий, производит их агрегацию и выдает агрегированные события на выходе.

Чтобы добавить агрегатор на конвейер:

1. Откройте окно конфигурации созданного конвейера. Убедитесь, что в выпадающем списке в правом верхнем углу окна выбран пункт **Черновик**.
2. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Агрегатор**. Отобразится окно добавления агрегатора.
3. Введите название агрегатора.
4. Добавьте в агрегатор правила агрегации.

   Чтобы добавить правило агрегации:

   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужное правило агрегации.

   b. Установите флажок слева от правила агрегации, которое необходимо добавить в агрегатор.
5. Нажмите на кнопку **Добавить**. Новый агрегатор отобразится на диаграмме конфигурации.

Пример:

Добавим агрегатор с правилом `NetworkActivityAggregationRule` на конвейер. В названии агрегатора укажем "AggregateEvents".

Настройка нормализации событий

Чтобы события, проходящие через конвейер обработки событий, нормализовывались, необходимо добавить нормализатор.

Чтобы добавить нормализатор, необходимо создать и установить на нем правило нормализации, согласно которому нормализатор будет обрабатывать и нормализовывать поступающие данные.

Создание правила нормализации

Правила нормализации в системе обрабатывают и преобразовывают поступающие события согласно условиям и коду нормализации, составленным на языке VRL и условию, заданному в поле `filter`.

Чтобы добавить правило нормализации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило нормализации**. Система отобразит окно создания правила нормализации.

▼ *Пример правила нормализации, схема RObject.*

```
# Уникальный идентификатор правила.
id: example/basic_normalization_rule

# Название правила.
name: BasicNormalizationRule

# Имя автора правила.
author: John Doe <johndoe@example.com>

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Базовое правило нормализации

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - logs/default.log
    - logs/somelog.log

# Теги, связанные с правилом.
tags:
  - normalization
  - security

# Тип элемента экспертизы.
type: normalization_rule

# Блок правил на языке VRL с условиями фильтрации событий.
filter: !vrl |
  .event.host == "170.174.42.191"

# Блок с правилами на языке VRL для нормализации полей события.
normalizer: !vrl |
    e = parse_json(.event) ?? {}
    . |= {
      "userId": e."user-identifier",
      "timestamp": parse_timestamp(e.datetime, "%d/%b/%Y:%H:%M:%S") ?? now(),
      "host": e.host,
      "method": e.method,
      "path": e.request,
      "protocol": e.protocol
    }

    if e.status == "404" || e.status == "401" {
      .responseStatus = "Bad request"
    }
```

```
    if e.protocol == "HTTP/2.0" {
        .protocol = "HTTP/2"
    }
```

▼ *Интерпретация*

Поле `filter`:

- `.event.host == "198.51.100.191"`:

  Сравнения поля `host` со значением "198.51.100.191" в сыром событии. Если значение поля `host` в сыром событии совпадает с "198.51.100.191", то событие будет нормализовано согласно правилам в поле `normalizer`. Если значение поля `host` не совпадает с указанным IP-адресом, правила нормализации к событию применены не будут.

Поле `normalizer`:

- `e = parse_json(.event) ?? {}`:

  Попытка преобразовать содержимое поля `.event` из формата JSON в структуру данных VRL.

  Если преобразование не удалось или результатом является `null`, переменной `e` будет присвоен пустой объект `{}`, что достигается с помощью оператора `??`.

- `. |= { ⋯ }`:

  Оператор обновления/добавления новых полей к текущему объекту на основе значений из объекта `e`.

Структура, которая присваивается текущему объекту:

- `userId`: присваивает значение поля `user-identifier` из объекта `e` к полю `userId` текущего объекта.
- `timestamp`: преобразует строковое значение поля `datetime` из объекта `e` в метку времени с использованием указанного формата. Если преобразование не удастся или значение поля `datetime` равно `null`, будет использовано текущее время, возвращаемое функцией `now()`.
- `host`: присваивает значение поля `host` из объекта `e` к полю `host` текущего объекта.
- `method`: присваивает значение поля `method` из объекта `e` к полю `method` текущего объекта.
- `path`: присваивает значение поля `request` из объекта `e` к полю `path` текущего объекта.
- `protocol`: присваивает значение поля `protocol` из объекта `e` к полю `protocol` текущего объекта.

`if e.status == "404" || e.status == "401"`:

- Условное выражение проверяет, равно ли значение поля `status` из объекта `e` строке "404" или "401".

`.responseStatus = "Bad request"`:

- Если условие выше выполняется, то поле `responseStatus` текущего объекта устанавливается в значение "Bad request".

`if e.protocol == "HTTP/2.0"`:

- Условное выражение проверяет, равно ли значение поля `protocol` из объекта `e` строке "HTTP/2.0".

`.protocol = "HTTP/2"`:

- Если условие выше выполняется, то поле `protocol` текущего объекта заменяется на значение "HTTP/2".

▼ *Обработка событий правилом нормализации*

Рассмотрим работу правила на примере разбора базового (сырого) события.

Допустим, на нормализатор поступают события следующего вида:

Пример 24. Исходное событие, JSON

```
{
  "event": "{\"host\":\"198.51.100.191\",\"user-
identifier\":\"JohnDoe\",\"datetime\":\"31/Jul/2023:06:55:49\",\"method\":\"GET\"
,\"request\":\"/observability/metrics/production\",\"protocol\":\"HTTP/2.0\",\"st
atus\":\"404\",\"bytes\":22427,\"referer\":\"https://example.url.com\"}"
}
```

После применения к событию правила "BasicNormalizationRule", оно принимает следующий вид:

Пример 25. Нормализованное событие, JSON

```
{
  "event": "{\"host\":\"198.51.100.191\",\"user-
identifier\":\"JohnDoe\",\"datetime\":\"31/Jul/2023:06:55:49\",\"method\":\"GET\"
,\"request\":\"/observability/metrics/production\",\"protocol\":\"HTTP/2.0\",\"st
atus\":\"404\",\"bytes\":22427,\"referer\":\"https://example.url.com\"}",
  "host": "170.174.42.191",
  "method": "GET",
  "path": "/observability/metrics/production",
  "protocol": "HTTP/2",
  "referer": "https://example.url.com",
  "responseBytes": 22427,
  "responseStatus": "Bad request",
  "serverIP": "198.51.100.191",
  "timestamp": "2023-07-30T23:55:49Z",
  "userAgent": null,
  "userId": "JohnDoe"
}
```

Событие удовлетворяет условию из поля `filter`, следовательно, оно подлежит нормализации и обрабатывается согласно правилам из поля `normalizer`. В структуре нормализованного

события в поле `event` записывается исходное событие. Прочие поля добавляются к объекту и определяются в соответствии с правилами нормализации, хранящимся в поле `normalizer`.

3. Заполните поля правила нормализации, чтобы определить логику его работы, и нажмите на кнопку **Опубликовать версию**. Система создаст правило и отобразит соответствующее уведомление. Новое правило нормализации отобразится в таблице раздела **Экспертиза**.
4. Чтобы созданное правило стало доступным на конвейерах событий, включите его:
   a. Нажмите на строку с созданным правилом нормализации. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
   b. Убедитесь, что в карточке открыта вкладка **Информация**.
   c. Переведите переключатель состояния в верхней части карточки в активное положение и подтвердите включение правила. Система обновит информацию о статусе правила нормализации.

Добавление нормализатора

Нормализатор представляет собой комплекс правил нормализации. Каждое такое правило состоит из двух основных компонентов: кода нормализации, выполненного на языке VRL, и фильтра. Если в правиле есть фильтр, то нормализация инициируется, только когда входящее событие соответствует условиям этого фильтра.

Порядок применения правил нормализации определяется последовательностью их добавления в нормализатор. Таким образом, события проходят через набор правил в установленном порядке, что позволяет осуществлять сложные сценарии нормализации данных.

Нормализатор на входе принимает события, преобразует и обогащает их, и выдает преобразованные события на выходе.

Чтобы добавить нормализатор на конвейер:

1. Откройте окно конфигурации созданного конвейера. Убедитесь, что в выпадающем списке в правом верхнем углу окна выбран пункт **Черновик**.
2. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Нормализатор**. Отобразится окно добавления нормализатора.
3. Введите название нормализатора.
4. При необходимости из выпадающего списка **Обработка ошибок** выберите шину коллектора, в которую будет отправлено событие в случае возникновения ошибок обработки. По умолчанию выбран вариант **События с ошибкой**.
5. Добавьте в нормализатор правила нормализации.

   Чтобы добавить правило нормализации:

   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужное правило нормализации.
   b. Установите флажок слева от правила нормализации, которое необходимо добавить в нормализатор.
6. Нажмите на кнопку **Добавить**. Новый нормализатор отобразится на диаграмме конфигурации.

Пример:

Добавим нормализатор с правилом `BasicNormalizationRule` на конвейер. В названии нормализатора укажем "Normalization".

Настройка корреляции событий

Чтобы события, поступающие на конвейер, коррелировались, необходимо добавить коррелятор.

Чтобы добавить коррелятор, необходимо создать и установить в нем правило корреляции. Правила корреляции определяют, как коррелятор будет обрабатывать входящие данные и формировать на их основе корреляционные события.

Создание правила корреляции

Правила корреляции в системе позволяют обнаруживать и анализировать взаимосвязанные события, соответствующие определенным условиям. Эти правила также могут инициировать действия после обнаружения, например, создание корреляционных событий.

Чтобы добавить правило корреляции:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило корреляции**. Система отобразит окно выбора способа создания правила.
3. Выберите способ создания правила через код и нажмите на кнопку **Создать**. Система отобразит окно задания RObject-конфигурации правила корреляции.

▼ *Пример правила корреляции, схема RObject*

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule_basic

# Название правила.
name: DDoSDetectionRule

# Тип элемента экспертизы.
type: correlation_rule

# Версия правила.
version: 1.0.0

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика

# Имя и контактные данные автора правила.
author: John Doe <johndoe@example.com>
```

```
# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .source_ip != null &&
  .request_path != null

# Разделение потоков событий на псевдонимы.
aliases:
  HighRateTraffic:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && parse_duration!(.window, "s") >= 100
  SuspiciousPatterns:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && ends_with(to_string!(.request_path),
"/api/highload")

# Соединение потоков событий.
select:
  alias: HighRateTraffic
  join:
    alias: SuspiciousPatterns
    absent: true
    on:
      - eq: { HighRateTraffic: .source_ip, SuspiciousPatterns: .source_ip }
    wait_for: 30

# Группировка событий.
group:
  - alias: HighRateTraffic
    by:
      - source_ip
    count: 3

# Частота генерации корреляционных событий в рамках каждой группы (событий в
секунду).
throttle_time_sec: 60

# Время хранения промежуточных корреляционных данных (секунды).
ttl: 60
```

```
# Логика обогащения корреляционного события на языке VRL.
on_correlate: !vrl |
  .message = "Обнаружена потенциальная DDoS атака с источника: " +
to_string!(%HighRateTraffic[0].source_ip)
  .additional_info = "Высокая частота запросов или подозрительные паттерны в пути
запроса /api/highload"

# Тесты для проверки работы правила.
tests:
  - name: DDoS Attack Detected
    events:
      - { "timestamp": "t'2025-01-19T20:10:44+00:00'", "event_type":
"HTTP_REQUEST", "source_ip": "192.0.2.1", "window": "120s", "request_path":
"start" }
      - { "timestamp": "t'2025-01-19T20:10:45+00:00'", "event_type":
"HTTP_REQUEST", "source_ip": "192.0.2.1", "window": "130s", "request_path":
"start" }
      - { "timestamp": "t'2025-01-19T20:10:46+00:00'", "event_type":
"HTTP_REQUEST", "source_ip": "192.0.2.1", "window": "140s", "request_path":
"start" }
      - { "timestamp": "t'2025-01-19T20:11:30+00:00'", "event_type":
"HTTP_REQUEST", "source_ip": "192.0.2.111", "window": "10s", "request_path":
"/api/highload" }
    assertion: !vrl |
      assert_eq!(.[0]message, "Обнаружена потенциальная DDoS атака с источника:
192.0.2.1")
```

▼ *Интерпретация*

Правило корреляции "DDoSDetectionRule" предназначено для выявления DDoS-атак на основе анализа потока поступающих событий.

Метаданные правила:

- `id`: уникальный идентификатор правила в системе.
- `name`: название правила.
- `type`: тип `correlation_rule` указывает на то, что это правило корреляции.
- `version`: версия правила.
- `description`: описывает цель правила — "Обнаружение атаки DDoS на основе трафика".
- `author`: создатель правила.
- `tags`: теги `ddos` и `security` классифицируют правило по его функциональности и области применения.

Логика фильтрации (filter):

- Условия фильтрации `.source_ip != null` и `.request_path != null` выбирают события, у которых есть IP-адрес источника и путь запроса.

Логика фильтрации (aliases):

- Псевдоним `HighRateTraffic`: фильтр для событий с высокой частотой запросов, идентифицируемых по `event_type == "HTTP_REQUEST"` и `parse_duration!(.window) >= 100`. Отбирает события с частотой запросов, превышающей или равной 100 запросов в определенный временной интервал.
- Псевдоним `SuspiciousPatterns`: фильтр для событий с подозрительными путями запроса, определяемый как `event_type == "HTTP_REQUEST"` и `ends_with(to_string!(.request_path), "/api/highload")`. Отбирает запросы, пути которых оканчиваются на "/api/highload".

Соединение потоков событий (select и join):

- Выбор и соединение (`select` и `join`) потоков событий `HighRateTraffic` и `SuspiciousPatterns` осуществляется на основе совпадения `source_ip`. Это соединение позволяет коррелировать высокочастотные запросы с подозрительными паттернами, исходящими от одного источника.
- В поле `wait_for` определяется, что система должна ожидать в течение 30 секунд перед тем, как принять решение о соединении событий из различных псевдонимов. Это значит, что система будет накапливать события в течение указанного времени с момента получения первого события и затем анализировать их на предмет соответствия заданным критериям соединения.
- В операции соединения с `SuspiciousPatterns` используется поле `absent: true`, что указывает на то, что отсутствие события из `SuspiciousPatterns` в течение 30 секунд будет добавлено в цепочку корреляционных событий.

Группировка событий (group):

- `group`: группировка событий применяется к псевдониму `HighRateTraffic` по полю `source_ip`, что позволяет агрегировать и анализировать события, исходящие от одного источника IP. Количество событий для формирования группы в поле `count` установлено в 3, что означает, что группа будет сформирована при накоплении 3 событий от одного источника IP.
- `throttle_time_sec`: задает интервал времени в секундах между генерацией корреляционных событий в рамках каждой группы событий. Например, 60 секунд.

Срок хранения промежуточных данных корреляции (ttl):

- `ttl` на уровне правила задает общее максимальное время жизни промежуточных корреляционных данных. Например, 60 секунд. Это время определяет, как долго промежуточные результаты для всей цепочки событий будут храниться в памяти до их автоматического удаления, если корреляция не завершается.

Пост-корреляционная логика (on_correlate):

- `on_correlate`: обогащает корреляционное событие полями `.message` и `.additional_info`.

По накоплении 3 событий правило генерирует корреляционное событие со следующими атрибутами:

- `type`: "correlation_rule",

- `event_type`: "DDoS Detection",
- `source_ip`: "192.168.1.1",
- `datetime`: "2023-09-28T12:45:00Z",
- `message`: "Обнаружена потенциальная DDoS атака с источника: 192.168.1.1",
- `additional_info`: "Высокая частота запросов и подозрительные паттерны в пути запроса /api/highload"

Тестирование правила (tests):

- Набор тестовых сценариев "DDoS Attack Detected" включает набор событий, часть из которых имитируют характеристики DDoS-атаки, а остальные являются контрольными примерами обычного трафика. Цель тестирования — подтвердить, что правило корректно идентифицирует только события, связанные с DDoS-атаками и игнорирует неподходящие события. Тесты включают события с HTTP-запросами, различающимися по `source_ip` и `request_path`, что позволяет проверить способность правила различать атакующий трафик от легитимного.
- События, моделирующие DDoS-атаку, имеют одинаковый `source_ip` и конечный путь запроса "/api/highload", что соответствует фильтрам `HighRateTraffic` и `SuspiciousPatterns`. Это сценарий, в котором ожидается активация правила и генерация корреляционного события.
- Контрольные события с разными `source_ip` и неподозрительными путями запроса ("/home", "/contact", "/about") предназначены для подтверждения, что правило не реагирует на обычные запросы, не связанные с DDoS-активностью.

`assertion` в тестовом блоке предназначен для проверки того, что только события, соответствующие характеристикам DDoS-атаки, приводят к формированию корреляционного события с ожидаемым сообщением об обнаружении атаки.

▼ *Обработка событий правилом корреляции*

Проанализируем логику работы правила "DDoSDetectionRule", фокусируясь на процессе обработки событий.

Предположим, что за короткий промежуток времени на коррелятор, на котором установлено правило "DDoSDetectionRule", поступает поток JSON-событий от различных источников IP, включая множество запросов с IP-адреса "192.168.1.1". Эти события соответствуют критериям, заданным в фильтрах `HighRateTraffic` и `SuspiciousPatterns`.

Пример 26. Исходное событие, JSON

```
{
  "event_type": "HTTP_REQUEST",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:34:56Z",
  "request_path": "/api/highload",
  "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/93.0.4472.124 Safari/537.36"
}
```

Здесь `x` в `source_ip` — уникальный номер для каждого события.

По мере поступления этих событий, они фильтруются правилом "DDoSDetectionRule". Фильтр `HighRateTraffic` отбирает события, соответствующие запросам с высокой частотой (более 100 запросов за заданный временной интервал), а `SuspiciousPatterns` фокусируется на запросах с определенным путем запроса `/api/highload`.

Когда события от одного источника IP, например, "192.168.1.1", соответствуют обоим фильтрам и проходят через механизм соединения `join`, правило "DDoSDetectionRule" приступает к группировке. Правило использует поле `group` для агрегирования событий от одного источника IP. Если количество событий от одного источника соответствует заданному порогу в 3 события, правило генерирует корреляционное событие.

Пример 27. Корреляционное событие, генерируемое правилом "DDoSDetectionRule"

```
{
  "type": "correlation_rule",
  "severity": "high",
  "event_type": "DDoS Detection",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:45:00Z",
  "message": "Обнаружена потенциальная DDoS атака с источника: 192.168.1.1",
  "additional_info": "Высокая частота запросов и подозрительные паттерны в пути
запроса /api/highload"
}
```

Данное корреляционное событие указывает на то, что система обнаружила аномальную активность, соответствующую критериям DDoS-атаки, исходящей с одного IP-адреса. Правило "DDoSDetectionRule" коррелирует события, основываясь на двух ключевых факторах: частоте запросов и специфических путях запросов, что является индикатором потенциальной атаки.

4. Заполните поля правила корреляции, чтобы определить логику его работы, и нажмите на кнопку **Опубликовать версию**. Система создаст правило и отобразит соответствующее уведомление. Новое правило корреляции отобразится в таблице раздела **Экспертиза**.
5. Чтобы созданное правило стало доступным на конвейерах событий, включите его:
   a. Нажмите на строку с созданным правилом корреляции. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
   b. Убедитесь, что в карточке открыта вкладка **Информация**.
   c. Переведите переключатель состояния в верхней части карточки в активное положение и подтвердите включение правила. Система обновит информацию о статусе правила корреляции.

В примере рассматривается способ создания правила корреляции через код. Для правила корреляции доступен также способ построения через специальный конструктор.

Добавление коррелятора

Коррелятор позволяет связывать различные события и создавать комплексные правила на их основе. На входе применяет правила корреляции к потоку событий. На выходе выдает корреляционные события.

Чтобы добавить коррелятор на конвейер:

1. Откройте окно конфигурации созданного конвейера. Убедитесь, что в выпадающем списке в правом верхнем углу окна выбран пункт **Черновик**.
2. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Коррелятор**. Отобразится окно добавления коррелятора.
3. Введите название коррелятора.
4. Задайте интервал очистки событий в секундах. По истечении указанного периода в императивных правилах корреляции будет происходить очистка событий с истекшим временем жизни. Время жизни задается в поле `expire_after_sec` в структуре правила.

   По умолчанию интервал очистки равен 300 секундам.

5. Добавьте в коррелятор правила корреляции.

   Если коллектор, на котором размещен коррелятор, настроен для работы в режиме распределенной корреляции, он не поддерживает работу императивных правил корреляции.

   Чтобы добавить правило корреляции:

   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужное правило корреляции.

   b. Установите флажок слева от правила корреляции, которое необходимо добавить в коррелятор.

6. Нажмите на кнопку **Добавить**. Новый коррелятор отобразится на диаграмме конфигурации.

Пример:

Добавим коррелятор с правилом DDoSDetectionRule на конвейер. Укажем в названии "DDoSDetectionRule".

Добавление конечной точки

Конечная точка определяет, куда направляются обработанные события, например, в хранилище или на внешний интерфейс. Конечная точка, в зависимости от настроек, пересылает нормализованное событие в хранилище событий или внешнюю систему. Конечная точка имеет один вход и не имеет выходов.

Для отладки конвейера вы можете использовать конечную точку Blackhole, которая не сохраняет и не отправляет события.

Чтобы добавить конечную точку на конвейер:

1. Откройте окно конфигурации созданного конвейера. Убедитесь, что в выпадающем списке в правом верхнем углу окна выбран пункт **Черновик**.
2. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Конечная точка**. Отобразится окно добавления конечной точки.
3. При необходимости выберите шаблон для автоматического заполнения полей конечной точки. По умолчанию выбран вариант **Без шаблона**.
4. Заполните поля (набор и содержание полей могут отличаться в зависимости от типа точки):
   - Название конечной точки.
   - Тип конечной точки.
   - Формат конечной точки.
5. Установите флажок **Сохранить как шаблон**, если настройку конечной точки необходимо сохранить в качестве шаблона для дальнейшего использования.
6. Нажмите **Добавить**. Новая конечная точка отобразится на диаграмме конфигурации.

Пример:

Создадим пять конечных точек со следующими названиями: "ServerError", "UnknownError", "HostUnavailable", "DDosDetections" и "198.51.100.191 Status".

Объединение элементов конвейера

Каждый элемент конвейера отвечает за конкретный этап обработки событий. Элемент имеет хотя бы один вход или выход, при этом вход может быть только один, а выходов может быть несколько. Входы и выходы элементов связаны между собой направляющими линиями, которые определяют порядок и направление обработки событий.

Чтобы приблизить или отдалить блоки на диаграмме конфигурации, используйте колесо мыши.

Чтобы связать элементы:

1. Наведите курсор на метку, расположенную на его боковой стороне. Курсор примет вид крестика.
2. Перетащите крестик к метке на другом элементе, чтобы установить связь. Порядок обработки событий будет определен логикой этих связей.

Пример настройки конвейера:

1. Соедините точку входа **JSON** с маршрутизатором **StatusCheckRouter**.
2. Свяжите точку входа **JSON** с агрегатором **AggregateEvents**.
3. Объедините агрегатор **AggregateEvents** с коррелятором **DDoSDetectionRule**.
4. Направьте поток событий из **DDoSDetectionRule** к конечной точке **DDosDetections**.
5. Направьте поток из маршрута **ClientErrorRoute** маршрутизатора к фильтру **FilterByHost**.
6. Направьте поток из маршрута **ServerErrorRoute** маршрутизатора к конечной точке **ServerError**.
7. Направьте поток из маршрута **По умолчанию** маршрутизатора к конечной точке **UnknownError**.

8. Направьте поток данных из **FilterByHost** к конечной точке **198.51.100.191 Status** и к VRL-трансформации **AssignResponseStatus**.
9. Направьте поток из **AssignResponseStatus** к нормализатору **Normalization**.
10. Соедините нормализатор **Normalization** с конечной точкой **HostUnavailable**.

Если нужно удалить связь, выберите ее и нажмите клавишу DELETE (на Windows можно также использовать BACKSPACE).

На этот конвейер события поступают через точку входа **JSON** и разделяются на две ветки: к маршрутизатору **StatusCheckRoute** и агрегатору **AggregateEvents**.

Агрегированные события попадают на коррелятор **DDoSDetectionRule**. Если коррелятор генерирует событие, оно сохраняется в конечную точку **DDosDetections**.

В маршрутизаторе **StatusCheckRoute** данные фильтруются по условиям маршрутов:

- **ClientErrorRoute**: `.event.status == "400" || .event.status == "401" || .event.status == "403" || .event.status == "404"`

  Соответствующие условию события двигаются к фильтру **FilterByHost**.

- **ServerErrorRoute**: `.event.status == "500" || .event.status == "502" || .event.status == "504"`

  Соответствующие условию события отправляются в конечную точку **ServerError**.

- **По умолчанию**: через этот маршрут проходят события, не удовлетворяющие условиям остальных маршрутов. Соответствующие события отправляются в конечную точку **UnknownError**.

В фильтре **FilterByHost** данные фильтруются по условию `.event.host == "198.51.100.191"`. Соответствующие события двигаются далее к конечной точке **198.51.100.191 Status**, где они записываются в хранилище, и к VRL-трансформации **AssignResponseStatus**, где им присваивается новое поле `responseStatus`.

События из **AssignResponseStatus** идут к нормализатору **Normalization**. На этапе нормализации поля события трансформируются. Если поле `status` имеет значение "Bad request", полю `severity` присваивается значение "high". Затем эти события отправляются в конечную точку **HostUnavailable**.

Включение коллектора и установка конфигурации конвейера

Для включения конвейера убедитесь, что связанный с ним коллектор уже включен.

Чтобы события начали поступать в систему и обрабатываться:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Убедитесь, что в карточке коллектора открыта вкладка **Информация**.
4. Переведите переключатель состояния коллектора в верхней части карточки в активное

положение. Отобразится окно подтверждения включения коллектора.

5. Нажмите на кнопку **Включить**. Система включит коллектор и отобразит соответствующее уведомление.
6. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
7. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
8. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации, над которой расположены кнопки установки/отключения конфигурации конвейера.
9. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Черновик**.
10. Нажмите на кнопку **Установить конфигурацию** над диаграммой конфигурации. Отобразится окно подтверждения установки конфигурации.
11. Нажмите на кнопку **Установить**. Система установит конфигурацию конвейера и отобразит соответствующее уведомление.

Теперь события начнут поступать в систему и обрабатываться согласно настройкам конвейера. Все события, которые проходят через конвейер или генерируются на нем, будут сохраняться согласно настройкам конечных точек. Поиск по этим событиям можно осуществлять с помощью языка запросов RQL в разделе **Поиск**.

Обработка событий на нескольких конвейерах

Данное руководство описывает процесс настройки обработки событий на нескольких конвейерах и создания шины для связи их друг с другом при поступлении событий из нескольких источников.

Об обработке событий на нескольких конвейерах

Обработка событий на основе нескольких конвейеров рекомендуется, когда данные собираются с нескольких источников. В этом случае настраивается конфигурация из нескольких взаимосвязанных конвейеров внутри одного коллектора, что упрощает отладку.

Для каждого источника данных создается конвейер, на котором настраивается сбор и нормализация событий. Итоговая корреляция всех нормализованных событий выполняется на отдельном конвейере. Конвейеры связаны друг с другом посредством шины.

Настройка конфигурации для обработки событий на основе нескольких конвейеров включает в себя следующие этапы:

1. Создание коллектора
2. Настройка шины
3. Добавление источников
    a. Создание конвейера
    b. Добавление точки входа
    c. Настройка обработки данных

i. Добавление маршрутизатора

ii. Нормализация данных

iii. Настройка агрегации событий

d. Настройка соединения с шиной

4. Настройка корреляции событий

a. Создание конвейера для корреляции

b. Настройка отправки данных из шины

c. Добавление коррелятора

d. Настройка конечных точек

5. Включение коллектора и установка конфигурации конвейеров

Создание коллектора

Чтобы создать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания коллектора.
3. Введите название коллектора.
4. Выберите подключение к пространству, в котором нужно создать коллектор. По умолчанию коллектор создается в центральном пространстве.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок + и — .

- При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов в каждом узле кластера Kubernetes.
- Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.
- Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

6. При необходимости введите описание коллектора.
7. Для работы в режиме распределенной корреляции переведите переключатель **Распределенный коррелятор** в активное положение.

Изменение этой настройки потребует перезапуска коллектора.

Если система работает на базе кластера Kubernetes с одним узлом, включение режима распределенной корреляции недоступно.

8. Укажите количество ядер процессора (**CPU**) и объем оперативной памяти (**RAM**), выделяемый для работы распределенного коррелятора. Это можно сделать непосредственно в поле или с помощью кнопок + и —.

Операции полей `join` и `group` декларативного правила вычисляются на коррелляторе. Остальные поля, в том числе `alias` и `filter`, вычисляются на предпроцессорах. В составе распределенного коррелятора может быть только один коррелятор, но несколько предпроцессоров.

   a. Укажите ресурсы для коррелятора:

      i. Выберите узел для размещения коррелятора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.

      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы коррелятора. Количество доступных в узле ресурсов отображается под полями ввода.

   b. Укажите ресурсы для предпроцессора:

      i. Выберите узел для размещения предпроцессора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.

      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы предпроцессора. Количество доступных в узле ресурсов отображается под полями ввода.

9. Нажмите на кнопку **Добавить предпроцессор**, если необходимо добавить дополнительный предпроцессор.
10. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:
    - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
    - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
    - **Info** — записывается общая информация о работе и состоянии коллектора.
    - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
    - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.
11. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее уведомление. Новый коллектор появится в списке раздела **Ресурсы → Коллекторы**.

Настройка шины

Шина позволяет направлять потоки событий между конвейерами. В коллекторе может быть одна или несколько шин. Шина поддерживает два типа соединения: на получение и на отправку событий. Два конвейера событий могут сообщаться друг с другом посредством шины.

Чтобы создать шину:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора, созданного на предыдущем этапе. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Шины** в карточке коллектора. Система отобразит список шин коллектора.
4. В нижней части карточки нажмите на кнопку **Добавить**. На экране отобразится окно добавления шины.
5. Введите название шины.
6. При необходимости введите описание шины.
7. Нажмите на кнопку **Добавить**. Шина будет создана и отобразится в списке шин на вкладке **Шины** в карточке коллектора.

Добавление источников

Настройте конвейеры для каждого источника данных, повторив следующие шаги.

Создание конвейера

Чтобы добавить конвейер:

1. Находясь в разделе **Ресурсы** → **Коллекторы**, нажмите на строку коллектора, созданного ранее. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
2. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
3. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
4. Введите название конвейера.
5. При необходимости введите описание конвейера.

Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

6. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке на вкладке **Конвейеры** в карточке коллектора.

Добавление точки входа

Точка входа поставляет в систему поток сырых событий. Выбор формата точки входа зависит от формата данных, получаемых из источника события.

Для отладки конвейера вы можете использовать точку входа Demo Logs, которая генерирует случайные события.

Чтобы добавить точку входа на конвейер:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Точка входа**. Отобразится окно добавления точки входа.
2. При необходимости выберите шаблон для автоматического заполнения полей точки входа. По умолчанию выбран вариант **Без шаблона**.
3. Заполните поля (набор и содержание полей могут отличаться в зависимости от типа точки):
    - Название точки входа.
    - Тип точки входа.
    - Политика аудита источников.

    Точка входа может быть привязана только к одной политике аудита источников, но к одной политике может быть привязано несколько точек входа.

    - Формат точки входа.
    - Данные для подключения.
    - Настройки шифрования соединения (опционально).
    - Количество событий (опционально).
    - Интервал в секундах (опционально).
4. Установите флажок **Сохранить как шаблон**, если настройку точки входа необходимо сохранить в качестве шаблона для дальнейшего использования.
5. Нажмите **Добавить**. Новая точка входа отобразится на диаграмме конфигурации.

Пример:

Добавим соединение с источником, который поставляет данные в формате системного журнала. Для этого выберем тип точки входа Syslog.

Настройка обработки данных

Чтобы привести события к модели, удобной для обогащения, корреляции и хранения, их нужно обработать с помощью нормализатора.

Данные из различных событий можно объединять согласно заданной логике. Для этого используется агрегатор.

Если события необходимо направлять на разные этапы обработки в зависимости от их содержимого, настройте условия фильтрации с помощью маршрутизатора.

Пример:

Рассмотрим сценарий обработки событий в зависимости от их формата.

Добавление маршрутизатора

Чтобы настроить маршрутизацию событий:

1. Добавьте на конвейер маршрутизатор и задайте в нем маршруты, фильтрующие события.

2. Соедините точку входа со входом маршрутизатора.

Пример:

Рассмотрим пример условия, которое выполняется, если в поле "appname" текста базового события записано значение "some-syslog".

Пример 28. Условие на VRL

```
.raw.appname == "some-syslog"
```

Нормализация данных

Чтобы настроить нормализацию данных:

1. Перейдите в раздел **Экспертиза** и создайте правило нормализации.

Если поток событий направляется по разным маршрутам, создайте несколько соответствующих правил нормализации.

2. Добавьте на конвейер нормализаторы, используя ранее созданные правила нормализации. Необходимо добавить на конвейер по одному нормализатору на каждый маршрут.
3. Соедините выходы маршрутизатора с входами нужных нормализаторов.

Пример:

Рассмотрим фрагмент правила нормализации.

Пример 29. Код VRL

```
...
.dvendor = "Linux"
.dproduct = "Some Product"
.dvchost = .raw.hostname
.msg = .raw.message
...
```

Настройка агрегации событий

Чтобы настроить агрегацию данных:

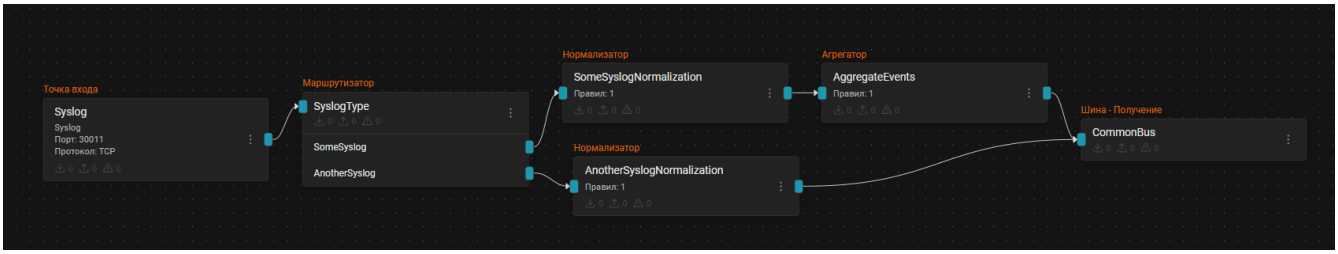
1. Перейдите в раздел **Экспертиза** и создайте правило агрегации.
2. Добавьте на конвейер агрегатор, используя ранее созданное правило агрегации.
3. Соедините выход соответствующего нормализатора со входом агрегатора.

Настройка соединения с шиной

Чтобы настроить соединение с шиной:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Шина**.

2. Выберите из выпадающего списка ранее созданную шину. На выбор доступны только шины текущего коллектора.
3. Введите название соединения с шиной. Выберите из выпадающего списка тип соединения **Получение**.
4. Нажмите на кнопку **Добавить**. Соединение с шиной отобразится на диаграмме конфигурации.
5. Соедините выходы элементов, в которых происходила предварительная обработка данных, со входом шины.



Настройка корреляции событий

Создание конвейера для корреляции

Добавьте конвейер, в котором будет происходить корреляция и отправка событий.

Настройка отправки данных из шины

Чтобы настроить отправку данных из шины:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Шина**.
2. Выберите из выпадающего списка ранее созданную шину. На выбор доступны только шины текущего коллектора.
3. Введите название соединения с шиной. Выберите из выпадающего списка тип соединения **Отправка**.
4. Нажмите на кнопку **Добавить**. Соединение с шиной отобразится на диаграмме конфигурации.

Добавление коррелятора

Чтобы настроить корреляцию данных:

1. Перейдите в раздел **Экспертиза** и создайте правило корреляции.
2. Добавьте на конвейер коррелятор, используя ранее созданное правило корреляции.
3. Соедините выход шины со входом коррелятора.

При необходимости добавьте на конвейер фильтры и прочие элементы для дополнительной обработки и анализа событий.

Настройка конечных точек

Конечная точка, в зависимости от настроек, пересылает событие в хранилище событий или внешнюю систему.

Чтобы настроить отправку событий:

1. Добавьте на конвейер одну или несколько конечных точек, в которых будут сохраняться события.
2. Соедините выходы из шины, коррелятора, созданных на предыдущих шагах, и/или из дополнительных элементов, со входами соответствующих конечных точек.

Включение коллектора и установка конфигурации конвейеров

Чтобы события начали поступать в коллектор и обрабатываться в соответствии с заданными правилами и условиями:

1. Перейдите в карточку коллектора и включите его.
2. Установите текущую конфигурацию на каждом конвейере.

Обработка событий на нескольких коллекторах

Данное руководство описывает процесс настройки обработки событий на нескольких коллекторах при поступлении событий из множественных источников.

Об обработке событий на нескольких коллекторах

Обработка событий на основе нескольких коллекторов рекомендуется в тех случаях, когда данные собираются с пяти и более источников. В этом случае настраивается конфигурация из нескольких взаимосвязанных конвейеров, которые распределены по нескольким коллекторам.

Для каждого источника данных создается отдельный коллектор, на котором настраивается сбор и нормализация событий. Итоговая корреляция данных выполняется на дополнительном коллекторе. Связи между коллекторами задаются через конечные точки и точку входа типа Vector.

Такая конфигурация удобна для настройки и масштабирования, а также обладает высокой отказоустойчивостью за счет независимости от изменений в одном из источников.

Настройка конфигурации для обработки событий на основе нескольких коллекторов включает в себя следующие этапы:

1. Добавление источников
    a. Создание коллектора и конвейера
    b. Добавление точки входа
    c. Настройка обработки данных
    d. Добавление конечной точки Vector
2. Настройка корреляции событий
    a. Создание коллектора и конвейера для корреляции
    b. Добавление точки входа Vector
    c. Добавление коррелятора
    d. Настройка конечных точек

3. Включение коллектора и установка конфигурации конвейеров

Добавление источников

Настройте коллекторы для каждого источника данных, повторив следующие шаги.

Если требуется получать данные на нескольких конвейерах в рамках одного коллектора, соедините их с помощью шины, как описано в статье Настройка отправки данных из шины. В отдельном конвейере соедините шину, настроенную на получение данных, с конечной точкой Vector.

Создание коллектора и конвейера

Чтобы создать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания коллектора.
3. Введите название коллектора.
4. Выберите подключение к пространству, в котором нужно создать коллектор. По умолчанию коллектор создается в центральном пространстве.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок + и —.

- При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов в каждом узле кластера Kubernetes.
- Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.
- Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

6. При необходимости введите описание коллектора.
7. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:
   - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
   - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
   - **Info** — записывается общая информация о работе и состоянии коллектора.
   - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.

- **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.

8. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее уведомление. Новый коллектор появится в списке раздела **Ресурсы** → **Коллекторы**.

Чтобы добавить конвейер:

1. Находясь в разделе **Ресурсы** → **Коллекторы**, нажмите на строку коллектора, созданного ранее. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
2. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
3. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
4. Введите название конвейера.
5. При необходимости введите описание конвейера.

Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

6. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке на вкладке **Конвейеры** в карточке коллектора.

Добавление точки входа

Точка входа поставляет в систему поток сырых событий. Выбор формата точки входа зависит от формата данных, получаемых из источника события.

Для отладки конвейера вы можете использовать точку входа Demo Logs, которая генерирует случайные события.

Чтобы добавить точку входа на конвейер:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Точка входа**. Отобразится окно добавления точки входа.
2. При необходимости выберите шаблон для автоматического заполнения полей точки входа. По умолчанию выбран вариант **Без шаблона**.
3. Заполните поля (набор и содержание полей могут отличаться в зависимости от типа точки):
   - Название точки входа.
   - Тип точки входа.
   - Политика аудита источников.

Точка входа может быть привязана только к одной политике аудита источников, но к одной политике может быть привязано несколько точек входа.

- Формат точки входа.
- Данные для подключения.
- Настройки шифрования соединения (опционально).
- Количество событий (опционально).
- Интервал в секундах (опционально).

4. Установите флажок **Сохранить как шаблон**, если настройку точки входа необходимо сохранить в качестве шаблона для дальнейшего использования.
5. Нажмите **Добавить**. Новая точка входа отобразится на диаграмме конфигурации.

Пример:

Добавим соединение с источником, который поставляет данные в формате системного журнала. Для этого выберем тип точки входа Syslog.

Настройка обработки данных

Чтобы привести события к модели, удобной для обогащения, корреляции и хранения, их нужно обработать с помощью нормализатора.

Чтобы настроить нормализацию данных:

1. Перейдите в раздел **Экспертиза** и создайте правило нормализации.

Если поток событий направляется по разным маршрутам, создайте несколько соответствующих правил нормализации.

2. Добавьте на конвейер нормализаторы, используя ранее созданные правила нормализации. Необходимо добавить на конвейер по одному нормализатору на каждый маршрут.
3. Соедините выходы маршрутизатора с входами нужных нормализаторов.

Пример:

Рассмотрим фрагмент правила нормализации.

Пример 30. Код VRL

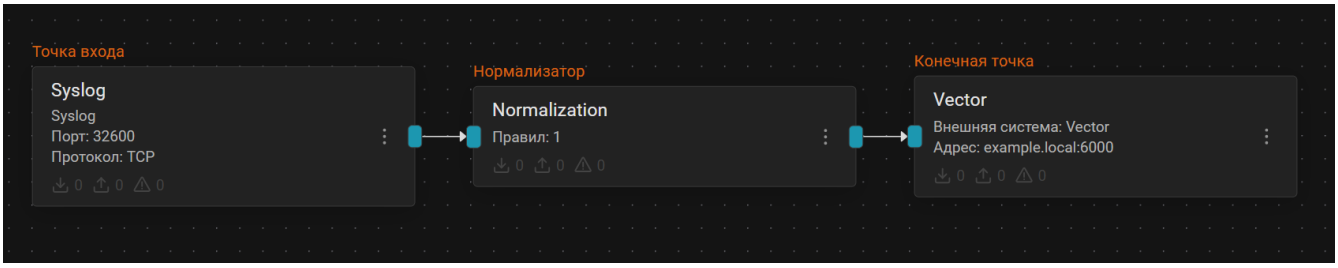
```
...
.dvendor = "Linux"
.dproduct = "Some Product"
.dvchost = .raw.hostname
.msg = .raw.message
...
```

При необходимости добавьте на конвейер маршрутизаторы, агрегаторы и другие элементы для дополнительной обработки событий.

Добавление конечной точки Vector

Чтобы настроить отправку данных из конвейера:

1. Добавьте на конвейер конечную точку, выбрав тип **Внешняя система** и тип внешней системы Vector.
2. Соедините выходы из нормализатора, созданного на предыдущем шаге, и/или из дополнительных элементов, со входом конечной точки.



Настройка корреляции событий

Создание коллектора и конвейера для корреляции

Чтобы создать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания коллектора.
3. Введите название коллектора.
4. Выберите подключение к пространству, в котором нужно создать коллектор. По умолчанию коллектор создается в центральном пространстве.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок + и —.

   - При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов в каждом узле кластера Kubernetes.
   - Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.
   - Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

6. При необходимости введите описание коллектора.
7. Для работы в режиме распределенной корреляции переведите переключатель **Распределенный коррелятор** в активное положение.

   Изменение этой настройки потребует перезапуска коллектора.

Если система работает на базе кластера Kubernetes с одним узлом, включение режима распределенной корреляции недоступно.

8. Укажите количество ядер процессора (**CPU**) и объем оперативной памяти (**RAM**), выделяемый для работы распределенного коррелятора. Это можно сделать непосредственно в поле или с помощью кнопок + и —.

Операции полей `join` и `group` декларативного правила вычисляются на коррелляторе. Остальные поля, в том числе `alias` и `filter`, вычисляются на предпроцессорах. В составе распределенного коррелятора может быть только один коррелятор, но несколько предпроцессоров.

   a. Укажите ресурсы для коррелятора:

      i. Выберите узел для размещения коррелятора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.

      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы коррелятора. Количество доступных в узле ресурсов отображается под полями ввода.

   b. Укажите ресурсы для предпроцессора:

      i. Выберите узел для размещения предпроцессора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.

      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы предпроцессора. Количество доступных в узле ресурсов отображается под полями ввода.

9. Нажмите на кнопку **Добавить предпроцессор**, если необходимо добавить дополнительный предпроцессор.

10. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:

    - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
    - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
    - **Info** — записывается общая информация о работе и состоянии коллектора.
    - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
    - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.

11. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее уведомление. Новый коллектор появится в списке раздела **Ресурсы → Коллекторы**.

12. Добавьте конвейер, в котором будет происходить корреляция и отправка событий.

Добавление точки входа Vector

Добавьте на конвейер точку входа, выбрав тип Vector.

Добавление коррелятора

Чтобы настроить корреляцию данных:

1. Перейдите в раздел **Экспертиза** и создайте правило корреляции.
2. Добавьте на конвейер коррелятор, используя ранее созданное правило корреляции.
3. Соедините выход из точки входа Vector со входом коррелятора.

Настройка конечных точек

Конечная точка, в зависимости от настроек, пересылает событие в хранилище событий или внешнюю систему.

Чтобы настроить отправку событий:

1. Добавьте на конвейер одну или несколько конечных точек, в которых будут сохраняться события.
2. Соедините выход коррелятора, созданного на предыдущем шаге, со входами конечных точек.

Включение коллектора и установка конфигурации конвейеров

Чтобы события начали поступать в коллектор и обрабатываться в соответствии с заданными правилами, повторите для каждого коллектора следующие действия:

1. Перейдите в карточку коллектора и включите его.
2. Установите текущую конфигурацию на конвейере.

Обработка событий с нескольких IP-адресов

Данное руководство описывает процесс настройки конвейера для обработки событий, поступающих на одну точку входа с нескольких IP-адресов.

Об обработке событий с нескольких IP-адресов

Система позволяет принимать в одну точку входа события с нескольких IP-адресов и настраивать для каждого IP-адреса отдельные правила обработки событий в рамках одного конвейера, без необходимости создания разных конвейеров с отдельными точками входа на каждый IP-адрес.

Для настройки обработки событий с разных IP-адресов понадобятся следующие элементы конвейера:

1. **Точка входа**: подключается к источнику данных и выдает поток исходных событий источника.

   Возможность определения IP-адреса доступна только для событий в точках входа Socket и Syslog.

2. **Маршрутизатор**: проверяет каждое событие по условиям маршрутов и направляет его на выходы тех маршрутов, где условие фильтрации выполняется.

   Каждый маршрут должен быть настроен на проверку нужного IP-адреса. Таким образом, события, пришедшие с одного IP-адреса, будут перенаправляться на один и тот же маршрут.

3. **Нормализаторы**: приводят структуру полученных событий к единому формату для упрощения последующей работы.

   На каждый маршрут должен быть настроен отдельный нормализатор.

4. **Конечные точки**: отправляют обработанные события в хранилища данных и внешние системы.
5. **Дополнительные элементы**: зависят от целей и задач пользователя по обработке и анализу событий.

Таким образом, настройка конвейера для обработки событий с нескольких IP-адресов включает следующие этапы:

1. Добавление конвейера
2. Настройка конфигурации конвейера
   a. Точка входа
   b. Маршрутизатор
   c. Нормализация данных
   d. Дополнительные элементы
   e. Конечные точки
3. Включение коллектора и установка конфигурации конвейера

Добавление конвейера

Конвейер — это упорядоченная последовательность взаимосвязанных элементов, выполняющих отдельные этапы обработки событий в системе.

Чтобы добавить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора, в котором будет выполняться настройка конвейера. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. В нижней части карточки нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
5. Введите название конвейера.
6. При необходимости введите описание конвейера.

Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в

окно настроек конфигурации конвейера после его создания.

7. Нажмите на кнопку **Добавить**. Система добавит конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке конвейеров на вкладке **Конвейеры** в карточке коллектора.

Настройка конфигурации конвейера

При настройке конвейера необходимо создать его компоненты, сконфигурировать их, упорядочить и объединить в последовательность, соответствующую логике обработки данных.

Чтобы перейти к настройке конфигурации конвейера:

1. Откройте карточку конвейера по стрелке (⌄) в его строке и нажмите на кнопку **Конфигурация конвейера** в нижней части карточки.
2. В открытом окне конфигурации конвейера убедитесь, что в выпадающем списке **Версия** в правом верхнем углу окна выбран пункт **Черновик**.

Точка входа

Точка входа — это элемент конвейера, который отвечает за доставку в систему исходных событий. Имеет один выход и не имеет входов.

Группировка поступающих событий по IP-адресу доступна только для точек входа следующих типов:

- Socket;
- Syslog.

Добавьте на конвейер точку входа. При создании выберите нужный тип точки входа и заполните ее параметры.

Маршрутизатор

Маршрутизатор — это элемент конвейера, который позволяет на основе заданных фильтров (маршрутов) направлять события на разные этапы обработки в зависимости от их содержимого или других условий. На входе маршрутизатора каждое событие проверяется по условиям маршрутов и появляется на выходах тех маршрутов, где условие выполняется.

Чтобы настроить маршрутизацию событий:

1. Добавьте на конвейер маршрутизатор и задайте в нем маршруты, фильтрующие события по IP-адресам.
2. Соедините выход из точки входа со входом маршрутизатора.

Настройки маршрутов

Для задания условий маршрутов по IP-адресам используются поля метаданных событий. Обращение к полям метаданных осуществляется через символ процента (%). Набор полей метаданных зависит от типа точки входа.

Таблица 2. Метаданные Socket

Поле	Тип	Описание	Пример
`socket.host`	String	IP-адрес источника события.	192.0.2.1

Таблица 3. Метаданные Syslog

Поле	Тип	Описание	Пример
`socket.host`	String	IP-адрес источника события.	192.0.2.1

Примеры условий маршрутов

1. Фильтрация событий по IP-адресу 192.0.2.1:

   Пример 31. Пример условия

   ```
   %socket.host == "192.0.2.1"
   ```

2. Фильтрация событий по IP-адресу 192.0.2.1 или 192.0.2.2:

   Пример 32. Пример условия

   ```
   %socket.host == "192.0.2.1" || %socket.host == "192.0.2.2"
   ```

Нормализация данных

На этом этапе поступающие события приводятся к единому формату.

Нормализатор преобразовывает события с помощью установленных на нем правил нормализации. Этот элемент конвейера принимает на вход события, проводит их нормализацию и на выходе предоставляет нормализованные события.

Чтобы добавить нормализатор, необходимо создать и установить на нем правило нормализации, согласно которому нормализатор будет обрабатывать поступающие на него данные и нормализовывать их.

Чтобы настроить нормализацию данных:

1. Перейдите в раздел **Экспертиза**.
2. Создайте правило нормализации.

Если события с разных IP-адресов необходимо нормализовать по-разному, создайте несколько соответствующих правил нормализации.

3. Откройте окно конфигурации конвейера, созданного ранее. Убедитесь, что в выпадающем списке в правом верхнем углу окна выбран пункт **Черновик**.
4. Добавьте на конвейер нормализаторы, используя ранее созданные правила нормализации. Необходимо добавить на конвейер по одному нормализатору на каждый маршрут.
5. Соедините выходы маршрутизатора с входами нужных нормализаторов.

Дополнительные элементы

При необходимости добавьте на конвейер прочие элементы для дополнительной обработки и анализа событий.

Конечные точки

Конечная точка, в зависимости от настроек, пересылает событие в хранилище событий или внешнюю систему. Имеет один вход и не имеет выходов.

Чтобы настроить отправку событий:

1. Добавьте на конвейер одну или несколько конечных точек, в которых будут сохраняться события.
2. Соедините выходы из нормализаторов или из дополнительных элементов со входами конечных точек.

Включение коллектора и установка конфигурации конвейера

Перед включением конвейера убедитесь, что связанный с ним коллектор уже включен.

Чтобы события начали поступать на конвейер и обрабатываться в соответствии с заданными правилами и условиями:

1. Перейдите в карточку коллектора и включите его.
2. Откройте конфигурацию конвейера и установите текущую конфигурацию.

Сбор событий из вынесенного коллектора

Данное руководство описывает процесс настройки сбора событий из коллектора, расположенного в сателлите, и их обработки в коллекторе в центральном пространстве системы.

О сборе событий из вынесенного коллектора

Система позволяет создавать коллекторы, расположенные в удаленных кластерах — сателлитах. Такие коллекторы называются **вынесенными**.

Сателлиты не обладают полным объемом функциональности системы. В частности, в сателлитах отсутствует возможность сохранять события в хранилище, так как хранилища событий подключаются к центральному пространству. Таким образом, чтобы отправлять на хранение события, собранные в вынесенном коллекторе, необходимо настроить их отправку из вынесенного коллектора в коллектор центрального пространства через глобальную шину.

Настройка конфигурации для сбора событий из вынесенного коллектора и их дальнейшей обработки в коллекторе центрального пространства системы включает следующие этапы:

1. Подключение к пространству сателлита
2. Настройка отправки событий из вынесенного коллектора

a. Создание вынесенного коллектора
b. Создание конвейера в вынесенном коллекторе
c. Добавление точки входа для сбора событий
d. Настройка отправки событий через глобальную шину

3. Настройка сбора и обработки событий в центральном пространстве
   a. Создание коллектора в центральном пространстве
   b. Создание конвейера в коллекторе центрального пространства
   c. Настройка получения событий из глобальной шины
   d. Настройка обработки событий
   e. Настройка отправки событий на хранение
4. Включение коллектора и установка конфигурации конвейеров

Подключение к пространству сателлита

Управление сателлитом осуществляется за счет запускаемых на нем сервисов через веб-интерфейс основного экземпляра системы. Чтобы подключиться к сателлиту, необходимо создать подключение к его пространству. Для этого:

1. Войдите в веб-интерфейс основного экземпляра системы.
2. Перейдите в раздел **Настройки** → **Пространства и сервисы** → **Пространства**.
3. Нажмите на кнопку **Создать** (┼). Отобразится окно создания подключения.
4. Введите идентификатор пространства сателлита в поле **ID**.
5. Введите название подключения. Название может быть любым.
6. В поле **IP или домен** введите IP-адрес или доменное имя (FQDN) одного из узлов кластера, в котором развернут сателлит.
7. В поле **Порт** введите номер TCP-порта, указанного на этапе **NATS satellite setup** установки сателлита.
8. При необходимости введите описание подключения к пространству.
9. Нажмите на кнопку **Создать**. Система создаст подключение к пространству и отобразит соответствующее уведомление. Подключение отобразится в списке раздела **Настройки** → **Пространства и сервисы** → **Пространства**.

У созданного подключения отобразится статус **Подключается**. По завершении подключения статус изменится на **Активен**.

Время на инициализацию подключения зависит загруженности кластеров и каналов связи и может достигать 5—10 минут.

Настройка отправки событий из вынесенного коллектора

Настройте отправку событий в вынесенном коллекторе, выполнив следующие шаги.

Создание вынесенного коллектора

Чтобы создать вынесенный коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (＋). Отобразится окно создания коллектора.
3. Введите название коллектора.
4. Выберите подключение к пространству, созданное на этапе Подключение к пространству сателлита текущей инструкции.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок ＋ и —.

- При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов в каждом узле кластера Kubernetes.
- Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.
- Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

6. При необходимости введите описание коллектора.
7. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:
   - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
   - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
   - **Info** — записывается общая информация о работе и состоянии коллектора.
   - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
   - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.
8. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее уведомление. Новый коллектор появится в списке раздела **Ресурсы** → **Коллекторы**.

Создание конвейера в вынесенном коллекторе

Чтобы добавить конвейер:

1. Находясь в разделе **Ресурсы** → **Коллекторы**, нажмите на строку коллектора, созданного на предыдущем шаге. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.

2. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
3. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
4. Введите название конвейера.
5. При необходимости введите описание конвейера.

Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

6. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке на вкладке **Конвейеры** в карточке коллектора.

Добавление точки входа для сбора событий

Точка входа поставляет в систему поток сырых событий. Выбор формата точки входа зависит от формата данных, получаемых из источника события.

Для отладки конвейера вы можете использовать точку входа Demo Logs, которая генерирует случайные события.

Чтобы добавить точку входа на конвейер:

1. Откройте конфигурацию конвейера, созданного на предыдущем шаге. Для этого:
   a. Находясь на вкладке **Конвейеры** в карточке коллектора, нажмите на кнопку ⌄ в строке конвейера. Отобразится карточка конвейера.
   b. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Точка входа**. Отобразится окно добавления точки входа.
4. При необходимости выберите шаблон для автоматического заполнения полей точки входа. По умолчанию выбран вариант **Без шаблона**.
5. Заполните поля (набор и содержание полей могут отличаться в зависимости от типа точки):
   - Название точки входа.
   - Тип точки входа.
   - Политика аудита источников.

Точка входа может быть привязана только к одной политике аудита источников, но к одной политике может быть привязано несколько точек входа.

   - Формат точки входа.

- Данные для подключения.
- Настройки шифрования соединения (опционально).
- Количество событий (опционально).
- Интервал в секундах (опционально).

6. Установите флажок **Сохранить как шаблон**, если настройку точки входа необходимо сохранить в качестве шаблона для дальнейшего использования.
7. Нажмите **Добавить**. Новая точка входа отобразится на диаграмме конфигурации.

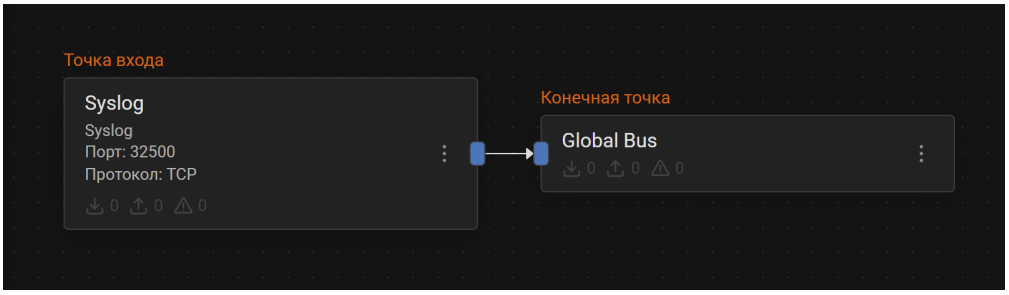
Пример:

Добавим соединение с источником, который поставляет данные в формате системного журнала. Для этого выберем тип точки входа Syslog.

Настройка отправки событий через глобальную шину

Чтобы отправлять события из вынесенного коллектора в коллектор центрального пространства, необходимо добавить конечную точку типа **Глобальная шина**. Для этого:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Конечная точка**. Отобразится окно добавления конечной точки.
2. Введите название конечной точки.
3. Выберите тип конечной точки **Глобальная шина**.
4. Нажмите на кнопку **Добавить**. Новая конечная точка отобразится в конфигурации конвейера.
5. Соедините выход из точки входа, созданной на предыдущем шаге, со входом конечной точки.



Настройка сбора и обработки событий в центральном пространстве

Настройте сбор событий из вынесенного коллектора в коллекторе центрального пространства, выполнив следующие шаги.

Создание коллектора в центральном пространстве

Чтобы создать коллектор в центральном пространстве:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (┼). Отобразится окно создания коллектора.
3. Введите название коллектора.

4. Выберите подключение к пространству **Центральное пространство**.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок + и —.

- При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов в каждом узле кластера Kubernetes.
- Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.
- Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

6. При необходимости введите описание коллектора.
7. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:
   - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
   - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
   - **Info** — записывается общая информация о работе и состоянии коллектора.
   - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
   - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.
8. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее уведомление. Новый коллектор появится в списке раздела **Ресурсы → Коллекторы**.

Создание конвейера в коллекторе центрального пространства

Чтобы добавить конвейер:

1. Находясь в разделе **Ресурсы → Коллекторы**, нажмите на строку коллектора, созданного на предыдущем шаге. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
2. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
3. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
4. Введите название конвейера.
5. При необходимости введите описание конвейера.

Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в

> окно настроек конфигурации конвейера после его создания.

6. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке на вкладке **Конвейеры** в карточке коллектора.

Настройка получения событий из глобальной шины

Чтобы получать события из вынесенного коллектора, добавьте на конвейер точку входа Global Bus. Для этого:

1. Откройте конфигурацию конвейера, созданного на предыдущем шаге. Для этого:
   a. Находясь на вкладке **Конвейеры** в карточке коллектора, нажмите на кнопку ⌄ в строке конвейера. Отобразится карточка конвейера.
   b. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
2. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
3. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Точка входа**. Отобразится окно добавления точки входа.
4. Введите название точки входа.
5. Выберите тип точки входа **Global Bus**.
6. Выберите коллектор, настроенный на этапе Настройка отправки событий из вынесенного коллектора текущей инструкции.
7. Нажмите на кнопку **Добавить**. Новая точка входа отобразится в конфигурации конвейера.

Настройка обработки событий

При необходимости добавьте на конвейер прочие элементы для дополнительной обработки и анализа событий.

Пример:

Добавим VRL-трансформацию со следующим кодом:

```
.dvendor = "Linux"
.dproduct = "Some Product"
.dvchost = .raw.hostname
.msg = .raw.message
```

Соединим выход точки входа со входом VRL-трансформации.

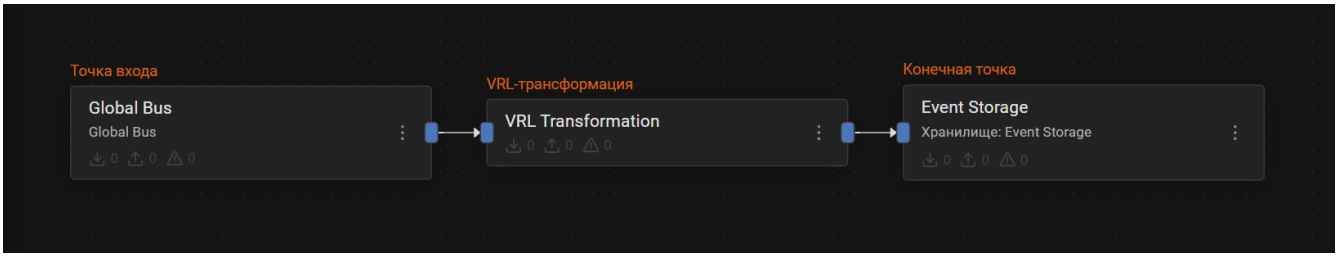
Настройка отправки событий на хранение

Чтобы отправить событие на хранение, необходимо добавить на конвейер конечную точку типа **Хранилище событий**. Для этого:

1. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Конечная**

точка. Отобразится окно добавления конечной точки.

2. Введите название конечной точки.
3. Выберите тип конечной точки **Хранилище событий**.
4. Выберите хранилище событий, в котором необходимо сохранять события.
5. Нажмите на кнопку **Добавить**. Новая конечная точка отобразится на диаграмме конфигурации.
6. Соедините выходы точки входа или дополнительных элементов со входом конечной точки.



Включение коллектора и установка конфигурации конвейеров

Чтобы события начали поступать в вынесенный коллектор и отправляться в коллектор центрального пространства, повторите для каждого коллектора следующие действия:

1. Перейдите в карточку коллектора и включите его.
2. Установите текущую конфигурацию на конвейере.

Автоматизация

Этот раздел предназначен для управления процессами автоматизации — триггерами и запусками.

Содержание раздела:

- Работа с триггерами
- Работа с запусками
- Работа с резолюциями

Работа с запусками

Данный раздел описывает процесс работы с запусками определенных элементов экспертизы в системе. Работа осуществляется в разделе **Автоматизация** → **Запуски** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О запусках

Запуск — это компонент системы, позволяющий запускать скрипты и плейбуки в системе, а также проверять статус их выполнения.

В систему можно добавлять следующие сущности, обеспечивающие автоматизацию процессов:

- скрипты — сценарии, определяющие последовательность выполнения различных действий.
- плейбуки — наборы связанных сценариев.

Условия автоматического запуска скриптов и плейбуков определяются триггерами. Триггеры срабатывают при наступлении определенных событий или выполнении определенных условий, которые в них заданы. При этом скрипты и плейбуки можно запустить не только по срабатыванию триггеров, но и вручную.

В разделе **Запуски** можно вручную запустить скрипты и плейбуки, а также просмотреть статусы всех выполняющихся и выполненных запусков.

Интерфейс раздела

Панель инструментов включает в себя следующие компоненты:

- Кнопка **Запустить** (▷) позволяет запустить выполнение скрипта или плейбука.
- Поле **Поиск** предназначено для быстрого поиска запусков по названию в отображаемом списке.
- Кнопка **Фильтр** позволяет настроить критерии фильтрации запусков. Запуски можно фильтровать по типу, времени старта и статусу.

В рабочей области отображается таблица имеющихся запусков. Таблица содержит следующие колонки:

- **Название** — название запуска.
- **Тип** — тип запуска (**Скрипт** или **Плейбук**).
- **Группа воркеров** — теги воркеров, на которых запущен скрипт или плейбук.
  - Если хотя бы один воркер с этим тегом находится в статусе **Включен**, отображается зеленый индикатор.
  - Если все воркеры с этим тегом находятся в статусе **Выключен** или **Ошибка**, отображается серый индикатор.
- **Время запуска** — дата и время старта запуска.
- **Время выполнения** — время, затраченное на выполнение запуска.
- **Статус** — статус запуска.
- **Триггер** — имя пользователя или название триггера, инициировавшего запуск.

  Информация, отображаемая в этой колонке, зависит от того, как именно был инициирован запуск:

  - имя пользователя — пользователь, запустивший скрипт или плейбук.
  - **Событие** — запуск скрипта или плейбука инициируется триггером типа **Событие триггера**.
  - **Расписание** — запуск скрипта или плейбука инициируется триггером типа **Расписание**.

  Событие триггера указывается в самом триггере. Расписание добавляется в параметры запуска или триггера.

  Если название триггера активно, по нажатию на него можно перейти в раздел **Автоматизация** → **Триггеры**.

При работе с таблицей запусков доступны следующие операции:

- Поиск запусков по полю **Название**.
- Фильтрация запусков по полям **Тип**, **Время запуска** и **Статус**:
  - Тип и статус запуска выбираются из выпадающих списков.
  - Начало и конец диапазона времени запуска задаются с помощью панели выбора даты и времени.
- Сортировка запусков по полям **Название**, **Время запуска**, **Статус** и **Триггер**.

Работа с запуском

Выполнение скриптов или плейбуков может инициироваться следующими триггерами:

- триггер типа **Расписание**. Триггер запускает скрипт или плейбук в соответствии с настройками расписания.
- триггер типа **Событие триггера**. Триггер запускает скрипт или плейбук при наступлении определенного события.

Доступные операции над запуском:

- Добавление запуска
- Просмотр запуска
- Остановка запуска

Добавление запуска

Запуск можно добавить, указав параметры вручную или выбрав ранее установленные параметры.

Чтобы добавить новый запуск:

1. Перейдите в раздел **Запуски**.
2. Нажмите на кнопку **Запустить** (▷). Отобразится форма добавления запуска.
3. Выберите тип элементов экспертизы для запуска — **Скрипт** или **Плейбук**.

   Для выбора доступны скрипты и плейбуки со статусом **Включен**.

4. Добавьте в запуск элемент экспертизы. Для этого:

   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужный элемент экспертизы.

   Панель инструментов включает в себя следующие компоненты:

   - Поле **Поиск** предназначено для быстрого поиска элементов экспертизы в отображаемом списке по названию и идентификатору.
   - Поле фильтра **Тип** позволяет выбрать тип элементов экспертизы для отображения.
   - Поле фильтра **Теги** позволяет задать комбинацию тегов искомых элементов.

   b. Выделите элемент экспертизы, который необходимо добавить в запуск.

   c. Нажмите на кнопку **Добавить**. Выбранный элемент экспертизы будет добавлен в запуск.

   После добавления выбранного элемента экспертизы в запуск открывается окно параметров запуска.

5. Укажите параметры запуска:

   a. В блоке **Настройки**:

      i. В поле **Название экспертизы** отображается элемент экспертизы, который будет запущен. Чтобы просмотреть элемент в отдельной вкладке, нажмите на кнопку **Открыть**.

      ii. Выберите из выпадающего списка ID тенанта, в котором будет выполнен запуск.

      По умолчанию в параметрах отображается текущий тенант, в котором работает пользователь.

      iii. Выберите из выпадающего списка **Группа воркеров** тег воркеров, на которых возможен запуск скрипта или плейбука.

В выпадающем списке отображаются теги имеющихся воркеров с их статусами. Для выполнения запуска хотя бы один из воркеров должен быть включен.

По умолчанию поле имеет значение `all`. В этом случае для запуска будет использоваться любой из доступных воркеров.

iv. В поле **ID интеграции** отображается идентификатор интеграции, позволяющий получить сведения о запуске для использования в скриптах и плейбуках. Эти сведения содержатся в специальном хранилище данных о запусках.

b. В блоке **Аргументы** при необходимости отредактируйте аргументы скрипта или плейбука, для которого добавляется запуск.

c. В блоке **Триггер** при необходимости укажите параметры триггера типа **Расписание** или **Событие триггера**, согласно которым будет производиться запуск. Для того чтобы указать параметры, необходимо перевести переключатель в верхней части блока **Триггер** в активное положение.

6. Нажмите на панели инструментов на кнопку **Запустить** (▷). Система добавит новый запуск для выбранного скрипта или плейбука и отобразит соответствующее уведомление.

Чтобы отменить добавление нового запуска и удалить введенные значения из полей, нажмите на кнопку **Отменить**.

Просмотр запуска

Чтобы просмотреть запуск:

1. Перейдите в раздел **Запуски**. Система отобразит сведения о всех запусках.
2. Нажмите на строку запуска в списке. Система отобразит в правой части экрана карточку с подробной информацией.

Название запуска отображается в верхней части карточки. Карточка содержит меню для действий с запуском:

- **Остановить** — остановка запуска.

Карточка содержит основные сведения о запуске:

- **ID тенанта** — идентификатор тенанта, создавшего запуск.
- **Название** — название используемого скрипта или плейбука.
- **Тип** — тип элементов экспертизы, используемый для запуска (скрипт или плейбук).
- **Группа воркеров** — список тегов воркеров, заданных для плейбука или скрипта.
  - Если хотя бы один воркер с этим тегом находится в статусе **Включен**, отображается зеленый индикатор.
  - Если все воркеры с этим тегом находятся в статусе **Выключен** или **Ошибка**, отображается серый индикатор.
- **Статус** — статус запуска.

Просмотреть ошибки запуска можно в карточке запуска по кнопке **Просмотреть логи**.

- **Триггер** — название триггера или имя пользователя (в зависимости от того, как был инициирован запуск).
- **Воркер** — название воркера, назначенного для выполнения задачи.
- **ID интеграции** — идентификатор интеграции.
- **Время запуска** — дата и время запуска.
- **Время выполнения** — время, затраченное на выполнение запуска.
- **Резолюция** (для плейбуков с модулем `approval` в статусе **На согласовании**) — ссылка **Открыть резолюцию** для перехода на страницу **Автоматизация → Резолюции**. На этой странице необходимо согласовать возобновление выполнения.

Таблица 4. Статусы элементов

Статус	Индикатор	Описание статуса
Выполнен	Зеленый	Задание выполнено без ошибок.
В процессе	—	Задание выполняется.
Ошибка	Красный	Задание не выполнилось или завершилось с ошибкой.
Согласован	Зеленый	Пользователь согласовал запрос. Выполнение задания возобновлено.
На согласовании	Синий	Схема выполняемого плейбука содержит модуль `approval`, требующий согласования дальнейшего выполнения. Система отправила пользователю запрос на согласование. Выполнение задания приостановлено и будет продолжено после того, как пользователь согласует запрос.
Не согласован	Желтый	Пользователь отклонил запрос на согласование. Выполнение задания остановлено.
Истекло время согласования	Желтый	У задания истекло время ожидания резолюции.

Остановка запуска

Чтобы остановить запуск:

1. Перейдите в раздел **Запуски**. Система отобразит сведения о запусках.
2. Нажмите на строку запуска в списке. Система отобразит страницу с карточкой этого запуска с подробной информацией о нем.

   В верхней части карточки отображается название запуска.

3. Нажмите на кнопку **Остановить**. Кнопка отображается только в карточках запусков в статусах **В очереди ожидания**, **На согласовании** и **В процессе**.

При остановке запуска некоторые операции могут быть выполнены частично, а часть информации утеряна.

4. Подтвердите остановку запуска. Запуск будет остановлен.

Статусы запуска

Таблица 5. Статусы запуска

Статус	Индикатор	Описание статуса
Выполнен	Зеленый	Плейбук выполнен.
В очереди ожидания	Синий	Плейбук находится в очереди ожидания на выполнение в воркере.
Согласован	Желтый	Пользователь согласовал запрос. Выполнение плейбука возобновлено.
В процессе	—	Плейбук выполняется.
Ошибка	Красный	Плейбук не выполнился или завершился с ошибкой.
Остановлен	Серый	Принудительная остановка выполнения плейбука.
На согласовании	Синий	Схема выполняемого плейбука содержит модуль `approval`, требующий согласования дальнейшего выполнения плейбука. Система отправила пользователю запрос на согласование. Выполнение плейбука приостановлено и будет продолжено после того, как пользователь согласует запрос.
Не согласован	Желтый	Пользователь отклонил запрос на согласование. Выполнение плейбука остановлено.
Истекло время согласования	Желтый	У плейбука истекло время ожидания резолюции.

Работа с триггерами

Данный раздел описывает процесс работы с триггерами определенных элементов экспертизы в системе. Работа осуществляется в разделе **Автоматизация** → **Триггеры** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О триггерах

Триггер — это компонент системы, определяющий в автоматическом режиме условия запуска элементов экспертизы (скриптов и плейбуков). Этими условиями являются наступление определенных событий или параметры расписания.

В системе можно задать два типа триггеров:

- **Расписание** — триггер, в котором условием запуска скрипта или плейбука являются параметры расписания.
- **Событие триггера** — триггер, в котором условием запуска скрипта или плейбука является наступление определенного события.

Интерфейс раздела

Панель инструментов включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новый триггер. По этой кнопке открывается форма создания триггера.
- Поле **Поиск** предназначено для быстрого поиска триггеров в отображаемом списке.
- Кнопка **Фильтр** позволяет настроить критерии фильтрации триггеров. В параметрах фильтрации доступны опции для установки **Типа** и **Статуса** искомых элементов.

В рабочей области отображается таблица имеющихся триггеров. При выборе конкретного триггера на экране открывается карточка с детальной информацией о нем.

Колонки таблицы представлены следующим образом:

- **Название** — название триггера.
- **Тип триггера** — тип события, инициирующего запуск скрипта или плейбука (**Расписание**, **Создание сущности**, **Изменение сущности**, **Удаление сущности**).
- **Детали** — сведения о расписании или событии, инициирующих запуск скрипта или плейбука.
  - Для расписания отображается cron-выражение, заданное в параметрах триггера.
  - Для события может отображаться следующая информация:
    - название события, которое происходит с сущностью (создание, изменение, удаление);
    - название **Пользовательский** — если пользователь самостоятельно определяет событие, которое послужит триггером.

- **Статус** — текущий статус триггера (**Включен** или **Выключен**).
- **Группа воркеров** — теги воркеров, используемых при запуске плейбука или скрипта.
  - Если хотя бы один воркер с этим тегом находится в статусе **Включен**, отображается зеленый индикатор.
  - Если все воркеры с этим тегом находятся в статусе **Выключен** или **Ошибка**, отображается серый индикатор.
- **Плейбук/Скрипт** — название плейбука или скрипта, запускаемого при выполнении условий триггера.
- **Дата создания** — дата и время создания триггера.
- **Дата изменения** — дата и время изменения триггера.

При работе с таблицей триггеров доступны следующие операции:

- Поиск триггеров по полям **Название** и **Плейбук/Скрипт**.
- Фильтрация триггеров по полям **Тип триггера** и **Статус**. Тип триггера и статус выбираются из выпадающих списков.
- Сортировка триггеров по полям **Название**, **Тип триггера**, **Статус**, **Дата создания** и **Дата изменения**.

Работа с триггером

Доступные операции над триггером:

- Создание триггера
- Просмотр триггера
- Включение и выключение триггера
- Изменение триггера
- Удаление триггера

Создание триггера

Создание триггера типа "Расписание"

Чтобы создать триггер типа "Расписание":

1. Перейдите в раздел **Автоматизация** → **Триггеры**.
2. Нажмите на кнопку **Создать** (＋). Отобразится окно создания триггера.
3. Добавьте элемент экспертизы. Для этого:
   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужный элемент экспертизы.

Панель инструментов окна добавления триггера включает в себя следующие компоненты:

- Поле **Поиск** предназначено для быстрого поиска элементов экспертизы

в отображаемом списке по названию и идентификатору.

- Поле фильтра **Тип** позволяет выбрать тип элементов экспертизы для отображения: **Скрипты** или **Плейбуки**.
- Поле фильтра **Теги** позволяет задать комбинацию тегов искомых элементов.

b. Выделите элемент экспертизы, который необходимо добавить в триггер.

c. Нажмите на кнопку **Добавить**. Выбранный элемент экспертизы будет добавлен в триггер.

После добавления выбранного элемента экспертизы в триггер открывается окно параметров запуска.

4. Укажите параметры запуска:

a. В блоке **Настройки**:

i. В поле **Название экспертизы** отображается элемент экспертизы, который будет запущен. Чтобы просмотреть элемент в отдельной вкладке, нажмите на кнопку **Открыть**.

ii. Выберите из выпадающего списка ID тенанта, в котором будет выполнен запуск.

По умолчанию в параметрах отображается текущий тенант, в котором работает пользователь.

iii. Выберите из выпадающего списка **Группа воркеров** тег воркеров, на которых возможен запуск скрипта или плейбука.

В выпадающем списке отображаются теги имеющихся воркеров с их статусами. Для выполнения запуска хотя бы один из воркеров должен быть включен.

По умолчанию поле имеет значение `all`. В этом случае для запуска будет использоваться любой из доступных воркеров.

iv. В поле **ID интеграции** отображается идентификатор интеграции, позволяющий получить сведения о запуске для использования в скриптах и плейбуках. Эти сведения содержатся в специальном хранилище данных о запусках.

b. В блоке **Триггер** выберите тип триггера для расписания и укажите параметры расписания, определяющего запуск триггера:

- В выпадающем списке **Тип** выберите значение **Расписание**.
- **Название** — название события триггера.
- **Описание** — описание события триггера.
- **Cron-выражение** — выражение, использующее синтаксис CRON и задающее время запуска.

Используются следующие символы:

- "*" — любое значение;
- "," — разделитель списка значений;

- "-" — диапазон значений;
- "/" — значение шага.

▼ *Примеры настроек Cron-выражения*

- Запускать каждые 20 минут с 8 до 18 с понедельника по пятницу:

```
0 */20 8-18 * * 1-5
```

- Запускать 1 января, 1 апреля, 1 июля и 1 октября в 2:00:00:

```
0 0 2 1 1,4,7,10 *
```

Ознакомьтесь с подробным описанием синтаксиса CRON.

- **Часовой пояс** — сведения о часовом поясе, для которого настраивается расписание. По умолчанию отображается часовой пояс системы.
- **Повторы запуска при ошибке**

  Значение в этом поле определяет, в каком режиме система будет запускать скрипт по этому триггеру — один раз или с повторением. В режиме повторения в случае выявления ошибок при запуске скрипт будет запускаться указанное количество раз и с указанным интервалом.

  В выпадающем списке доступны значения:

  - **Без повторов** — скрипт по триггеру будет запущен только один раз. Значение отображается по умолчанию.
  - **Постоянно** — скрипт по триггеру будет запускаться повторно в соответствии с параметрами повторного запуска.

    Если вам нужно, чтобы в случае ошибки запуска скрипта система пыталась запустить скрипт повторно, выберите из выпадающего списка значение **Постоянно** и укажите параметры повторного запуска:

  - **Количество попыток** — максимальное количество попыток запуска скрипта.
  - **Повторять через** — интервал между попытками запуска скрипта.
  - **Единицы измерения** — единицы измерения интервала запуска скрипта (сек., мин., ч., дн.).
- **Окончание действия расписания** — дата и время окончания действия расписания. В выпадающем списке доступны значения:
  - **Нет конечной даты** — расписание действует всегда (значение установлено по умолчанию).
  - **По дате** — расписание действует до указанной даты.

5. Нажмите на кнопку **Запустить** (▷). Триггер будет запущен. Система отобразит соответствующее уведомление.

Чтобы отменить добавление нового триггера и удалить введенные значения из полей, нажмите на кнопку **Отменить**.

Создание триггера типа "Событие триггера"

Чтобы создать триггер типа "Событие триггера":

1. Перейдите в раздел **Автоматизация** → **Триггеры**.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания триггера.
3. Добавьте элемент экспертизы. Для этого:
   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужный элемент экспертизы.

      Панель инструментов окна добавления триггера включает в себя следующие компоненты:

      - Поле **Поиск** предназначено для быстрого поиска элементов экспертизы в отображаемом списке по названию и идентификатору.
      - Поле фильтра **Тип** позволяет выбрать тип элементов экспертизы для отображения: **Скрипты** или **Плейбуки**.
      - Поле фильтра **Теги** позволяет задать комбинацию тегов искомых элементов.

   b. Выделите элемент экспертизы, который необходимо добавить в триггер.
   c. Нажмите на кнопку **Добавить**. Выбранный элемент экспертизы будет добавлен в триггер.

      После добавления выбранного элемента экспертизы в триггер открывается окно параметров запуска.

4. Укажите параметры запуска:
   a. В блоке **Настройки**:
      i. В поле **Название экспертизы** отображается элемент экспертизы, который будет запущен. Чтобы просмотреть элемент в отдельной вкладке, нажмите на кнопку **Открыть**.
      ii. Выберите из выпадающего списка ID тенанта, в котором будет выполнен запуск.

      По умолчанию в параметрах отображается текущий тенант, в котором работает пользователь.

      iii. Выберите из выпадающего списка **Группа воркеров** тег воркеров, на которых возможен запуск скрипта или плейбука.

      В выпадающем списке отображаются теги имеющихся воркеров с их статусами. Для выполнения запуска хотя бы один из воркеров должен быть включен.

      По умолчанию поле имеет значение `all`. В этом случае для запуска будет использоваться любой из доступных воркеров.

      iv. В поле **ID интеграции** отображается идентификатор интеграции, позволяющий получить сведения о запуске для использования в скриптах и плейбуках. Эти сведения содержатся

в специальном хранилище данных о запусках.

b. В блоке **Триггер** выберите тип триггера для события и укажите параметры события, наступление которого инициирует действие триггера:

- В выпадающем списке **Тип** выберите значение **Событие триггера**.
- **Название** — название события триггера.
- **Описание** — описание события триггера (опционально).
- **Тип события триггера** — тип события триггера.

  Доступные типы:

  - **Создание объекта**;
  - **Изменение объекта**;
  - **Удаление объекта**;
  - **Пользовательский** — событие, которое послужит триггером, задается пользователем.
    - Если выбран тип **Пользовательский**, укажите **модуль**, который создает события триггера, и **событие**, при наступлении которого произойдет запуск плейбука или скрипта.

      Список событий, которые можно использовать как триггер, доступен в хранилище событий аудита.

      Чтобы отобразить список событий и узнать модуль и название нужного события:

      I. Перейдите в раздел **Поиск** основного меню системы.

      II. Выберите хранилище событий аудита из списка хранилищ в верхней части экрана. Система отобразит список событий аудита в хранилище.

      III. Выберите из списка требуемое событие. Название модуля события указано в колонке **moduleKey**, а название события — в колонке **eventType**. Эти данные необходимо указать в форме создания триггера типа **Пользовательский**.

    - Если выбран любой другой тип события, выберите из списка **Домен** домен, в котором должно наступить событие, а затем из списка **Сущность** выберите сущность, для которой создается действие триггера.

- **Повторы запуска при ошибке** — это поле определяет, в каком режиме система будет запускать скрипт по триггеру — один раз или с повторением. В режиме повторения в случае выявления ошибок при запуске скрипт будет запускаться указанное количество раз и с указанным интервалом.

  В выпадающем списке доступны значения:

  - **Без повторов** — скрипт по триггеру будет запущен только один раз. Значение отображается по умолчанию.
  - **Постоянно** — скрипт по триггеру будет запускаться повторно в соответствии с параметрами повторного запуска.

    Если вам нужно, чтобы в случае ошибки запуска скрипта система пыталась запустить

скрипт повторно, выберите из выпадающего списка значение **Постоянно** и укажите параметры повторного запуска:

- **Количество попыток** — максимальное количество попыток запуска скрипта.
- **Повторять через** — интервал между попытками запуска скрипта.
- **Единицы измерения** — единицы измерения интервала запуска скрипта (сек., мин., ч., дн.).

5. Нажмите на кнопку **Запустить** (▷). Триггер будет запущен. Система отобразит соответствующее уведомление.

Чтобы отменить добавление нового триггера и удалить введенные значения из полей, нажмите на кнопку **Отменить**.

Просмотр триггера

Чтобы просмотреть триггер:

1. Перейдите в раздел **Автоматизация** → **Триггеры**. Система отобразит сведения об имеющихся триггерах.
2. Нажмите на строку триггера в списке. Система отобразит справа карточку с подробной информацией об этом триггере.

Название триггера отображается в верхней части карточки. Справа от названия расположены кнопки для изменения ( ) и удаления ( ) триггера.

Карточка триггера содержит вкладки:

- **Информация** — основные сведения о триггере.
- **Запуски** — информация о запусках триггера.

Вкладка Информация

Вкладка **Информация** содержит следующие поля:

- **ID тенанта** — ID тенанта, в котором был добавлен триггер.
- **Название** — название триггера.
- **Описание** — описание триггера.
- **Тип события** — тип события, инициирующего запуск скрипта или плейбука (**Расписание**, **Создание сущности**, **Изменение сущности**, **Удаление сущности**).
- **Группа воркеров** — теги воркеров, используемых при запуске скрипта или плейбука.
  - Если хотя бы один воркер с этим тегом находится в статусе **Включен**, отображается зеленый индикатор.
  - Если все воркеры с этим тегом находятся в статусе **Выключен** или **Ошибка**, отображается серый индикатор.
- **Плейбук/скрипт** — название плейбука или скрипта, запуск которого инициируется триггером.
- **ID интеграции** — идентификатор интеграции, позволяющий получить сведения о запуске для

использования в скриптах и плейбуках. Эти сведения содержатся в специальном хранилище данных о запусках.

Также вкладка содержит переключатель, позволяющий включить/выключить триггер. Если отключить триггер, запланированные запуски не будут выполнены.

Если среди включенных воркеров с тегом, заданным для скрипта или плейбука, нет воркеров со статусом **Активен** или **Ошибка обновления**, система отобразит предупреждение.

Ниже отображаются текущие настройки триггера:

- Для типа триггера **Расписание** в карточке отображаются настройки расписания.
- Для типа триггера **Событие триггера** в карточке отображаются настройки события.

Если ни один воркер с тегом, заданный для скрипта или плейбука, не включен, система отобразит предупреждение.

Вкладка Запуски

Вкладка **Запуски** содержит таблицу с информацией о запусках текущего триггера, в случае, если запуски были. Данные о запуске хранятся 30 дней с момента запуска. Таблица содержит следующие столбцы:

- **Время старта** — дата и время запуска триггера.
- **Время выполнения** — время, затраченное на выполнение запуска.
- **Статус** — статус запуска.

Сортировка списка запусков может осуществляться по следующим столбцам:

- **Время запуска**.
- **Статус**.

Включение и выключение триггера

Чтобы включить триггер:

1. Перейдите в раздел **Автоматизация** → **Триггеры**. Система отобразит сведения об имеющихся триггерах.
2. Нажмите на строку триггера в списке. Система отобразит справа карточку с подробной информацией об этом триггере.
3. Убедитесь, что в карточке открыта вкладка **Информация**.
4. Переведите переключатель в верхней части вкладки в активное положение. Отобразится окно подтверждения.
5. Нажмите на кнопку **Включить**. Система обновит статус триггера и отобразит соответствующее уведомление.

Чтобы выключить триггер:

1. Перейдите в раздел **Автоматизация** → **Триггеры**. Система отобразит сведения об имеющихся триггерах.
2. Нажмите на строку триггера в списке. Система отобразит справа карточку с подробной информацией об этом триггере.
3. Убедитесь, что в карточке открыта вкладка **Информация**.
4. Переведите переключатель в верхней части вкладки в неактивное положение. Отобразится окно подтверждения.
5. Нажмите на кнопку **Выключить**. Система обновит статус триггера и отобразит соответствующее уведомление.

Если выключить триггер, запланированные запуски не будут выполнены.

Изменение триггера

Чтобы изменить триггер:

1. Перейдите в раздел **Автоматизация** → **Триггеры**. Система отобразит сведения об имеющихся триггерах.
2. Нажмите на строку триггера в списке. Система отобразит справа карточку с подробной информацией об этом триггере.
3. Нажмите на кнопку **Изменить** ( ) в верхней части карточки триггера. Отобразится окно настроек триггера.
4. Внесите изменения в требуемые поля триггера.
5. Нажмите на кнопку **Сохранить**. Измененные данные триггера будут сохранены.

Панель инструментов окна изменения триггера включает в себя следующие компоненты:

- **Сохранить** позволяет сохранить измененные параметры триггера.
- **Отменить** позволяет отменить изменение параметров триггера и удалить введенные значения из полей.

Удаление триггера

1. Перейдите в раздел **Автоматизация** → **Триггеры**. Система отобразит сведения об имеющихся триггерах.
2. Нажмите на строку триггера в списке. Система отобразит справа карточку с подробной информацией об этом триггере.
3. Нажмите на кнопку **Удалить** ( ) в верхней части карточки триггера. Отобразится окно подтверждения удаления триггера.
4. Нажмите на кнопку **Удалить**. Система отобразит уведомление об удалении выбранного триггера, и триггер исчезнет из списка триггеров.

Работа с резолюциями

Данный раздел описывает процесс работы с запросами, поступившими пользователю на

согласование в ходе работы плейбука. Пользователь изучает запросы и принимает по ним положительные или отрицательные решения (резолюции). До принятия решения выполнение плейбука приостанавливается. В зависимости от решения выполнение плейбука возобновляется или прекращается.

Работа осуществляется в разделе **Автоматизация** → **Резолюции** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О резолюциях

При создании схемы плейбуков можно использовать модули типа `approval`. Модуль этого типа позволяет пользователю возобновить выполнение плейбука только после согласования в системном интерфейсе запроса, поступившего из плейбука. До тех пор, пока пользователь не согласует этот запрос, выполнение плейбука приостанавливается.

Интерфейс раздела

Если в системе имеются запросы от плейбуков в статусе **На согласовании**, (т.е. запросы, требующие резолюции пользователя), рядом с названием раздела в главном меню отображается актуальное количество таких запросов.

Панель инструментов включает в себя следующие компоненты:

- Кнопка **Согласовать** позволяет согласовать поступивший запрос.
- Кнопка **Отклонить** позволяет отклонить поступивший запрос.

Доступны массовые операции согласования и отклонения запросов.

Кнопки становятся доступны, если в таблице запросов выбран хотя бы один запрос.

В рабочей области отображается таблица имеющихся запросов. При выборе конкретного запроса на экране открывается карточка с детальной информацией о нем.

Колонки таблицы представлены следующим образом:

- **Название** — название плейбука, от которого поступил запрос.
- **Описание** — описание запроса, поступившего на согласование. Задается в схеме плейбука.
- **Время ожидания** — интервал времени, в течение которого от пользователя ожидается согласование запроса. Задается в схеме плейбука.
- **Статус** — статус запроса. Доступный статус:
  - **На согласовании** — запрос требует резолюции пользователя.
- **Источник** — отображается имя пользователя, запустившего плейбук, или тип события, инициирующего запуск.
- **Дата создания** — дата создания запроса.

При работе с таблицей запросов доступны следующие операции:

- Поиск запросов по полю **Название**.
- Сортировка запросов по полям **Название**, **Время ожидания**, **Статус** и **Дата создания**.

Работа с запросом

Доступные операции над запросом:

- согласование запроса;
- отклонение запроса;
- просмотр запроса.

Согласование запроса

Чтобы согласовать запрос:

1. Перейдите в раздел **Автоматизация** → **Резолюции**. Система отобразит сведения об имеющихся запросах.
2. Нажмите на строку запроса в списке. Система отобразит справа карточку этого запроса.
3. Нажмите на кнопку **Согласовать** в нижней части карточки.

Вы также можете согласовать одновременно несколько запросов, выбрав их в списке и нажав на кнопку **Согласовать** над таблицей запросов.

Запрос можно согласовать только в рамках временного интервала, заданного в модуле `approval` схемы плейбука. Если время истекло, а запрос не согласован, выполнение плейбука будет прервано.

Отклонение запроса

Чтобы отклонить запрос:

1. Перейдите в раздел **Автоматизация** → **Резолюции**. Система отобразит сведения об имеющихся запросах.
2. Нажмите на строку запроса в списке. Система отобразит страницу с карточкой этого запроса.
3. Нажмите на кнопку **Отклонить** в нижней части карточки. Система отобразит окно с подтверждением отклонения запроса.
4. При необходимости укажите в этом окне причину отклонения запроса.
5. Нажмите на кнопку **Отклонить** в окне с подтверждением.

Вы также можете отклонить одновременно несколько запросов, выбрав их в списке и нажав на кнопку **Отклонить** над таблицей запросов.

Просмотр запроса

Чтобы просмотреть запрос:

1. Перейдите в раздел **Автоматизация** → **Резолюции**. Система отобразит сведения об имеющихся

запросах.

2. Нажмите на строку запроса в списке. Система отобразит карточку этого запроса с информацией о нем.

Название плейбука, от которого поступил запрос, отображается в верхней части карточки.

Чтобы закрыть карточку и вернуться к списку запросов, нажмите на крестик в правом верхнем углу карточки.

Карточка включает в себя следующие поля:

- **Название** — название плейбука, от которого поступил запрос.
- **Описание** — описание запроса, поступившего на согласование. Задается в схеме плейбука.
- **Время ожидания** — оставшееся время, в течение которого пользователь может согласовать запрос.
- **Статус** — статус запроса. Возможные статусы:
  - **На согласовании** — запрос требует резолюции пользователя.
- **Источник** — отображается имя пользователя, запустившего плейбук, или тип события, инициирующего запуск.
- **Дата создания** — дата создания запроса.

В нижней части карточки отображаются следующие кнопки:

- **Согласовать** — позволяет согласовать запрос.
- **Отклонить** — позволяет отклонить запрос.

Аналитика

В данном разделе описаны методы работы с инструментами аналитики данных в системе.

Содержание раздела:

- Работа с дашбордами
- Работа с отчетами
- Работа с разделом Поиск

Работа с дашбордами

Данный раздел описывает процесс работы с виджетами и дашбордами в системе. Работа осуществляется в разделе **Дашборды** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О дашбордах

Дашборды — специальные панели, предназначенные для представления информации о поступающих в систему событиях в графическом виде.

Чтобы визуализировать данные, на дашборд должны быть добавлены **виджеты**, позволяющие отображать эти данные с помощью графиков и диаграмм.

Интерфейс раздела

В верхней части раздела отображается название текущего дашборда. Выбрать другой дашборд можно, нажав на стрелку справа от названия. Система отобразит выпадающий список доступных дашбордов.

Пользователю доступны общие дашборды и созданные им персональные дашборды. Персональные дашборды обозначены меткой **Персональный**. Метка отображается на панели инструментов раздела, а также в выпадающем списке дашбордов в верхней части раздела.

Справа от названия дашборда отображаются следующие кнопки:

- Кнопка + позволяет добавить новый дашборд.
- Кнопка ↓ позволяет импортировать дашборды в систему.

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Добавить виджет** (+) позволяет создать новый виджет на текущем дашборде.
- Кнопка ✎ позволяет изменить текущий дашборд.

- Кнопка позволяет удалить текущий дашборд.
- Кнопка позволяет экспортировать дашборд в формате JSON.
- Кнопка позволяет сохранить дашборд как схему.
- Метка **Персональный** отображается, если дашборд доступен только текущему пользователю.
- Описание дашборда (при наличии) предоставляет краткую информацию о его функциональности.
- Кнопка **Переменные** ({;}) позволяет открыть боковую панель для задания переменных, которые можно использовать в RQL-запросах виджетов.
- Кнопка предназначена для расширения панели дашбордов на всю страницу браузера.

В рабочей области отображаются виджеты выбранного дашборда. Можно перемещать виджеты по рабочей области и изменять их размеры. В верхней части виджета отображается его наименование, а также кнопки:

- — для обновления данных виджета.
- — для изменения или удаления виджета, а также скрытия или отображения его легенды.

Работа с дашбордами

Доступные операции над дашбордами:

- Добавление дашборда
- Просмотр дашборда
- Расширение дашборда на полный экран
- Задание переменных
- Изменение дашборда
- Настройка доступности дашборда
- Удаление дашборда
- Импорт дашборда
- Экспорт дашборда
- Сохранение дашборда как схемы

Добавление дашборда

Создать дашборд можно следующими способами:

- Добавить дашборд и виджеты вручную.
- Использовать предварительно созданную схему.

Создание дашборда вручную

Чтобы создать дашборд вручную:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на кнопку **Добавить дашборд** в верхней части экрана. Отобразится окно создания

дашборда.

3. Введите название дашборда.
4. При необходимости введите описание дашборда.
5. Установите флажок **Персональный**, если дашборд необходимо сделать доступным только для текущего пользователя.
6. Нажмите на кнопку **Добавить**. Система создаст новый дашборд и отобразит соответствующее уведомление. Дашборд отобразится в выпадающем списке в верхней части раздела **Дашборды**. Система автоматически откроет добавленный дашборд для просмотра.

После создания дашборда добавьте в него необходимые виджеты.

Создание дашборда из схемы

Чтобы создать дашборд из схемы:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на кнопку **Добавить дашборд** в верхней части экрана. Отобразится окно создания дашборда.
3. Выберите из выпадающего списка схему дашборда.

Если до выбора схемы были заполнены название и описание дашборда, они будут заменены на данные из схемы.

4. Сопоставьте переменные виджетов с источниками информации — хранилищем событий или активным списком в зависимости от типа виджетов.
5. При необходимости введите название и описание дашборда вместо заполненных значений из схемы.
6. Установите флажок **Персональный**, если дашборд необходимо сделать доступным только для текущего пользователя.
7. Нажмите на кнопку **Добавить**. Система создаст новый дашборд и отобразит соответствующее уведомление. Дашборд отобразится в выпадающем списке в верхней части раздела **Дашборды**. Система автоматически откроет добавленный дашборд для просмотра.

Просмотр дашборда

Чтобы просмотреть дашборд:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.

Расширение дашборда на полный экран

Чтобы раскрыть дашборд на всю страницу браузера:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку в правом верхнем углу раздела. Содержимое дашборда будет расширено на всю страницу браузера.

   В расширенном режиме из верхней части раздела и с панели инструментов скрываются все компоненты, кроме выпадающего списка для выбора дашборда и кнопки для возвращения в обычный режим.

В расширенном режиме кнопка меняется на и предназначена для возвращения дашборда в обычный режим отображения.

Задание переменных

Переменные можно использовать в RQL-запросах при построении виджетов событий и виджетов активных списков.

Чтобы задать переменные:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку **Переменные** ({;}) в верхней части раздела. Система отобразит в правой части экрана панель настройки переменных.
5. Задайте аргументы и значения переменных.

   Чтобы добавить переменную:

   a. Нажмите на кнопку **Добавить**. Появится строка настройки переменной.
   b. Введите аргумент переменной. Аргумент может содержать только латинские буквы (A–Z, a–z) и цифры (0–9).
   c. Выберите тип значения переменной. Доступные типы:
      - **Строка**;
      - **Число**;
      - **Дата**;
      - **Логическое**.
   d. Укажите значение переменной. Способ задания значения зависит от его типа:
      - **Строка** и **Число** — введите значение вручную.
      - **Дата** — введите значение вручную или задайте с помощью панели выбора даты и времени.
      - **Логическое** — выберите значение из выпадающего списка: **True** или **False**.

Чтобы удалить переменную, нажмите на кнопку в строке ее настроек.

6. Нажмите на кнопку **Применить**, чтобы применить заданные переменные и перестроить дашборд с учетом их значений.

   Примененные переменные отобразятся в верхней части рабочей области раздела.

Чтобы сбросить примененную переменную, нажмите на кнопку ✕ справа от нее в верхней части рабочей области.

Чтобы сбросить сразу все переменные, выполните одно из действий:

- Нажмите на кнопку **Сбросить все** (✕) в верхней части рабочей области.
- Нажмите на кнопку **Сбросить все** в нижней части панели настроек переменных.

Изменение дашборда

Чтобы изменить дашборд:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку в верхней части раздела. Отобразится окно изменения дашборда.
5. Внесите изменения в требуемые поля.
6. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление.

Настройка доступности дашборда

Только пользователь, создавший дашборд, может изменить его доступность.

Чтобы настроить доступность дашборда:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку в верхней части раздела. Отобразится окно изменения дашборда.
5. Настройте доступность дашборда:
   - Чтобы сделать дашборд доступным только для текущего пользователя, установите флажок **Персональный**.
   - Чтобы сделать дашборд доступным для всех пользователей системы, снимите флажок **Персональный**.
6. Нажмите на кнопку **Сохранить**. Система сохранит изменения настроек и отобразит соответствующее уведомление. Обновленная информация отобразится на панели инструментов.

Удаление дашборда

Чтобы удалить дашборд:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку **Удалить дашборд** в верхней части раздела. Отобразится окно подтверждения удаления дашборда.
5. Нажмите на кнопку **Удалить**. Система удалит дашборд и отобразит соответствующее уведомление. Удаленный дашборд будет исключен из выпадающего списка дашбордов. Система автоматически откроет первый дашборд из списка на просмотр.

Вместе с дашбордом также удаляются и все добавленные на него виджеты.

Импорт дашборда

Чтобы импортировать дашборд:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на кнопку **Импорт** ( ) в верхней части экрана. Отобразится окно импорта дашборда.
3. Нажмите на кнопку **Выбрать файл** и выберите файл дашборда для импорта. Для импорта доступны только файлы формата JSON размером не более 10 МБ.
4. Нажмите на кнопку **Импортировать**. Система импортирует дашборд и отобразит соответствующее уведомление. Новый дашборд появится в выпадающем списке дашбордов.

Если вы импортируете дашборд с виджетом событий, который ссылается на недоступное хранилище, то вместо данных виджет отобразит ошибку о том, что хранилище не найдено.

Экспорт дашборда

Чтобы экспортировать дашборд:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку **Экспорт** ( ) в верхней части экрана. Отобразится окно экспорта дашборда.
5. Нажмите на кнопку **Экспортировать**. Система загрузит данные дашборда на устройство пользователя в формате JSON и отобразит соответствующее уведомление.

Сохранение дашборда как схемы

Чтобы сохранить дашборд как схему:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку **Схема** ( ) в верхней части экрана. Отобразится окно создания схемы.

Чтобы создать схему на основе дашборда, дашборд должен содержать хотя бы один виджет.

5. По умолчанию название и описание создаваемой схемы идентичны названию и описанию дашборда. При необходимости измените название и описание схемы.
6. Выберите из выпадающего списка каталог, в котором будет создана схема.
7. Для виджетов событий и активных списков автоматически генерируются переменные, которые при создании дашборда из схемы нужно связать с источниками данных. При необходимости измените переменные виджетов.

Чтобы при создании дашборда несколько виджетов получали данные из одного источника, задайте для них одинаковые переменные.

8. Нажмите на кнопку **Далее**. Откроется окно создания схемы с заполненными данными.
9. Задайте идентификатор схемы дашборда в поле `id`.
10. При необходимости внесите изменения в поля схемы.
11. Нажмите на кнопку **Опубликовать версию**. Система создаст схему дашборда и отобразит соответствующее уведомление. Новая схема появится в таблице раздела **Экспертиза**.

Чтобы схема дашборда стала доступной для создания дашбордов, ее необходимо включить.

Вы можете создать черновик схемы дашборда без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик схемы и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Работа с виджетами

Доступные операции над виджетами:

- Добавление виджета
- Обновление виджета
- Изменение размеров виджета
- Перемещение виджета
- Изменение настроек виджета
- Настройка отображения легенды виджета
- Настройка отображения рядов данных виджета

- Просмотр деталей ряда данных виджета
- Создание копии виджета
- Удаление виджета
- Экспорт виджета

Добавление виджета

Пользователь может создавать виджеты следующих типов:

- **Виджеты метрик** — отображают сведения об основных метриках системы.
- **Виджеты событий** — отображают сведения о событиях в системе на основе RQL-запроса.
- **Виджеты активных списков** — отображают сведения об активных списках на основе RQL-запроса.

Добавление виджета метрик

Чтобы добавить виджет метрик:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку **Добавить виджет** в верхней части раздела. Отобразится окно создания виджета.
5. Введите название виджета.
6. При необходимости введите описание виджета.
7. Выберите **Метрики** из выпадающего списка **Тип виджета**.
8. Выберите метрику для виджета из выпадающего списка:
   - **Хост БД CPU** — график использования ресурсов процессора (CPU) контейнерами БД.
   - **Хост БД RAM** — график использования ресурсов оперативной памяти (RAM) контейнерами БД.
   - **Хост Коллектор CPU** — график использования ресурсов процессора (CPU) контейнерами коллектора.
   - **Хост Коллектор RAM** — график использования ресурсов оперативной памяти (RAM) контейнерами коллектора.
   - **Пайплайн EPS** — график событий в секунду (EPS).
   - **Пайплайн объем трафика** — график объема трафика, поступающего в систему через конвейер.
   - **Пайплайн ErrPS** — график ошибок в секунду (ErrPS).
9. Установите временной диапазон для виджета:
   a. Задайте количество единиц времени вручную или с помощью кнопок — и +.
   b. Выберите из выпадающего списка единицы времени.

10. Если вы хотите, чтобы данные на виджете регулярно обновлялись, воспользуйтесь переключателем **Включить автообновление** и укажите периодичность обновления:
    a. Задайте частоту обновления вручную или с помощью кнопок — и +.
    b. Выберите из выпадающего списка единицы времени периода.
11. Чтобы предварительно просмотреть, как будут отображаться данные на виджете в соответствии с заданными настройками, нажмите на кнопку **Обновить** (🔄) в правой верхней части окна создания виджета.
12. Нажмите на кнопку **Добавить виджет**. Система создаст виджет и отобразит соответствующее уведомление. Новый виджет появится на дашборде.

Добавление виджета событий

Чтобы добавить виджет событий:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку **Добавить виджет** в верхней части раздела. Отобразится окно создания виджета.
5. Введите название виджета.
6. При необходимости введите описание виджета.
7. Выберите **События** из выпадающего списка **Тип виджета**.
8. Выберите хранилище событий, из которого требуется брать информацию о событиях, из выпадающего списка.
9. Выберите тип визуализации для виджета из выпадающего списка. Доступные типы:
    - Столбчатая диаграмма.
    - График.
    - Круговая диаграмма.
    - Столбчатая диаграмма с накоплением.
    - Таблица.
    - Итоги.
10. Задайте лимит строк для виджетов вручную или с помощью кнопок — и +. Лимит строк определяет максимальное количество строк, которые будут использованы для построения виджета. Допустимый диапазон лимита строк: 1 — 100 000.

    По умолчанию для таблиц и круговых диаграмм установлен лимит в 100 строк, для остальных типов визуализации — 1000 строк.

Лимит строк также можно задать в RQL-запросе с помощью оператора `LIMIT`. В этом случае лимит определяется минимальным из значений поля **Лимит строк** и оператора `LIMIT`. Например:

- Если в поле **Лимит строк** задано значение `100`, а в запросе указано `LIMIT 20`, то будет установлен лимит в 20 строк.
- Если в поле **Лимит строк** задано значение `20`, а в запросе указано `LIMIT 100`, то также будет установлен лимит в 20 строк.

Если задан лимит более 6 строк, то на диаграммах типов **График**, **Столбчатая диаграмма** и **Столбчатая диаграмма с накоплением** будет отображаться только часть подписей, с равномерными пропусками, чтобы избежать наложения.

11. Установите временной диапазон для виджета:
    a. Задайте количество единиц времени вручную или с помощью кнопок — и +.
    b. Выберите из выпадающего списка единицы времени.
12. Введите запрос на языке RQL в поле запроса.

    При необходимости вы можете вставить RQL-запрос из поискового шаблона. Для этого выберите нужный шаблон из выпадающего списка **Шаблон запроса**. Для быстрого поиска шаблона в выпадающем списке начните вводить его название в строке поиска.

    При необходимости вы можете задать переменные, которые можно использовать в RQL-запросе, в секции **Переменные**.

- Указанные переменные также будут применены к дашборду.
- Если заданная переменная уже существует на панели переменных дашборда, то ее значение на панели будет заменено на заданное в виджете.

    Чтобы добавить переменную:

    a. Нажмите на кнопку **Добавить**. Появится строка настройки переменной.
    b. Введите аргумент переменной. Аргумент может содержать только латинские буквы (`A–Z`, `a–z`) и цифры (`0–9`).
    c. Выберите тип значения переменной. Доступные типы:
        - **Строка**;
        - **Число**;
        - **Дата**;
        - **Логическое**.
    d. Укажите значение переменной. Способ задания значения зависит от его типа:
        - **Строка** и **Число** — введите значение вручную.
        - **Дата** — введите значение вручную или задайте с помощью панели выбора даты и времени.
        - **Логическое** — выберите значение из выпадающего списка: **True** или **False**.

    Чтобы удалить переменную, нажмите на кнопку в строке ее настроек.

Чтобы использовать указанную переменную в RQL-запросе, необходимо использовать шаблон Handlebars:

- Шаблоны для обращения к переменной типа **Строка**, **Число** или **Логическое**:
  - `{{<variable_argument>}}` — обращение к переменной с экранированием специальных символов в ее значении;
  - `{{{<variable_argument>}}}` — обращение к переменной без экранирования специальных символов в ее значении.
- Шаблоны для обращения к переменной типа **Дата**:
  - `{{sqlDateTime <variable_argument>}}` — обращение к значению типа DateTime64;
  - `{{sqlDateTime <variable_argument> false}}` — обращение к значению типа DateTime.

Здесь:

- `<variable_argument>` — аргумент переменной.

Если аргумент переменной содержит специальные символы или `0`, то при обращении к переменной необходимо экранировать ее аргумент одинарными кавычками, например: `{{'<variable_argument>'}}`.

Пример использования переменных в RQL-запросе

Предположим, что нам необходимо получить события, отфильтрованные по значениям `evo.siem` и `evo.core` поля `moduleKey`.

Чтобы передавать конкретные значения в RQL-запрос, можно создать две переменные:

- Первая переменная:
  - Аргумент: `moduleKey1`.
  - Тип значения: **Строка**.
  - Значение: `evo.siem`.
- Вторая переменная:
  - Аргумент: `moduleKey2`.
  - Тип значения: **Строка**.
  - Значение: `evo.core`.

Тогда для решения поставленной задачи можно использовать следующий запрос:

```
SELECT id, moduleKey, tenantId WHERE moduleKey IN
("{{moduleKey1}}","{{moduleKey2}}") ORDER BY tenantId
```

13. Укажите параметры в секции **Сопоставление полей**.

При использовании группировки на виджетах отображаются только первые 20 групп.

Набор доступных параметров зависит от выбранного типа диаграммы:

- **Столбчатая диаграмма** и **Столбчатая диаграмма с накоплением**: доступны поля **Легенда**, **Значение** и **Группировка** (последнее поле обязательно для типа **Столбчатая диаграмма с накоплением**).

  Столбчатая диаграмма позволяет отображать значения показателей в виде вертикальных столбцов. Высота столбцов пропорциональна значениям.

  Столбчатая диаграмма с накоплением позволяет сравнивать сгруппированные показатели с другими группами тех же показателей, например, распределение количества событий по IP-адресу источника с их группировкой по типу. Столбцы значений в каждой группе расположены друг поверх друга, формируя единый столбец в рамках группы.

  Пример

  Необходимо построить диаграмму, показывающую распределение количества событий по IP-адресу источника с их группировкой по типу.

  Запрос:

  ```
  SELECT sourceIp, type, count(id) as cnt GROUP BY sourceIp, type
  ```

  В поле **Легенда** указывается атрибут, возвращающий IP-адрес источника; в поле **Значение** указывается атрибут, возвращающий количество событий; в поле **Группировка** указывается атрибут, возвращающий тип события.

- **Круговая диаграмма**: доступны поля **Легенда** и **Значение**.

  Пример

  Необходимо построить диаграмму, показывающую распределение количества событий по ID коллекторов.

  Запрос:

  ```
  SELECT count(*) as cnt, collectorId GROUP BY collectorId
  ```

  В поле **Легенда** указывается атрибут, возвращающий ID коллектора; в поле **Значение** указывается атрибут, возвращающий количество событий.

- **График**: доступны поля **Ось X**, **Ось Y** и **Группировка**.

  Пример

Необходимо построить график, показывающий зависимость количества событий от ID коллекторов.

Запрос:

```
SELECT collectorId, count(*) AS cnt WHERE collectorId like '%'
GROUP BY collectorId
```

В поле **Ось X** указывается `collectorId` — атрибут, возвращающий ID коллектора; в поле **Ось Y** указывается `cnt` — псевдоним атрибута, возвращающего количество событий.

- **Итоги**: доступно поле **Значение**.

Пример

Необходимо вывести количество событий за указанный в окне редактирования виджета период.

Запрос:

```
SELECT count(id) AS cnt, toStartOfHour(timestamp) as timestamp
GROUP BY timestamp
```

В поле **Значение** указывается `cnt` — псевдоним атрибута, возвращающего количество событий.

- **Таблица**: для каждого столбца таблицы доступны по два поля: **Название колонки** и **Атрибут из запроса**.

Для добавления столбца в таблицу предназначена кнопка **Добавить колонку**, по которой появляется строка настроек нового столбца. Для удаления столбца из таблицы предназначена кнопка в строке его настроек.

Пример

Необходимо создать таблицу с двумя столбцами, позволяющую просмотреть соотношение количества событий для 10 последних коллекторов.

Запрос:

```
SELECT collectorId, count(*) AS cnt WHERE collectorId like '%'
GROUP BY collectorId
```

Название колонки	Атрибут из запроса
Коллектор	`collectorId` — атрибут, возвращающий ID коллектора

Название колонки	Атрибут из запроса
Количество событий	`cnt` — псевдоним атрибута, возвращающего количество событий

В поле **Лимит строк** указывается значение `10`, что позволяет вывести в таблицу не более 10 записей.

Лимит строк также можно указать в запросе с помощью оператора `LIMIT`:

```
SELECT collectorId, count(*) AS cnt WHERE collectorId like '%'
GROUP BY collectorId LIMIT 10
```

В этом случае лимит строк в таблице определяется минимальным из значений поля **Лимит строк** и оператора `LIMIT`.

14. Если вы хотите, чтобы данные на виджете регулярно обновлялись, воспользуйтесь переключателем **Включить автообновление** и укажите периодичность обновления:
    a. Задайте частоту обновления вручную или с помощью кнопок — и +.
    b. Выберите из выпадающего списка единицы времени периода.
15. Чтобы предварительно просмотреть, как будут отображаться данные на виджете в соответствии с заданными настройками, нажмите на кнопку **Обновить** (🔄) в правой верхней части окна создания виджета. Доступны два варианта предпросмотра:
    a. **Диаграмма** — отображает визуальное представление виджета на дашборде.
    b. **Данные** — отображает результаты RQL-запроса в табличной форме.
16. Нажмите на кнопку **Добавить виджет**. Система создаст виджет и отобразит соответствующее уведомление. Новый виджет появится на дашборде.

Добавление виджета активного списка

Чтобы добавить виджет активного списка:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку **Добавить виджет** в верхней части раздела. Отобразится окно создания виджета.
5. Введите название виджета.
6. При необходимости введите описание виджета.
7. Выберите **Активный список** из выпадающего списка **Тип виджета**.
8. Выберите активный список, из которого требуется брать информацию о записях.
9. Выберите тип визуализации для виджета из выпадающего списка. Доступные типы:

- Столбчатая диаграмма.
- График.
- Круговая диаграмма.
- Столбчатая диаграмма с накоплением.
- Таблица.
- Итоги.

10. Задайте лимит строк для виджетов вручную или с помощью кнопок — и +. Лимит строк определяет максимальное количество строк, которые будут использованы для построения виджета. Допустимый диапазон лимита строк: 1 — 100 000.

    По умолчанию для таблиц и круговых диаграмм установлен лимит в 100 строк, для остальных типов визуализации — 1000 строк.

    Лимит строк также можно задать в RQL-запросе с помощью оператора `LIMIT`. В этом случае лимит определяется минимальным из значений поля **Лимит строк** и оператора `LIMIT`. Например:

    - Если в поле **Лимит строк** задано значение `100`, а в запросе указано `LIMIT 20`, то будет установлен лимит в 20 строк.
    - Если в поле **Лимит строк** задано значение `20`, а в запросе указано `LIMIT 100`, то также будет установлен лимит в 20 строк.

    Если задан лимит более 6 строк, то на диаграммах типов **График**, **Столбчатая диаграмма** и **Столбчатая диаграмма с накоплением** будет отображаться только часть подписей, с равномерными пропусками, чтобы избежать наложения.

11. Установите временной диапазон для виджета:
    a. Задайте количество единиц времени вручную или с помощью кнопок — и +.
    b. Выберите из выпадающего списка единицы времени.
12. Введите запрос на языке RQL в поле запроса.

    При необходимости вы можете вставить RQL-запрос из поискового шаблона. Для этого выберите нужный шаблон из выпадающего списка **Шаблон запроса**. Для быстрого поиска шаблона в выпадающем списке начните вводить его название в строке поиска.

    При необходимости вы можете задать переменные, которые можно использовать в RQL-запросе, в секции **Переменные**.

    - Указанные переменные также будут применены к дашборду.
    - Если заданная переменная уже существует на панели переменных дашборда, то ее значение на панели будет заменено на заданное в виджете.

    Чтобы добавить переменную:

a. Нажмите на кнопку **Добавить**. Появится строка настройки переменной.

b. Введите аргумент переменной. Аргумент может содержать только латинские буквы (`A–Z`, `a–z`) и цифры (`0–9`).

c. Выберите тип значения переменной. Доступные типы:

   - **Строка**;
   - **Число**;
   - **Дата**;
   - **Логическое**.

d. Укажите значение переменной. Способ задания значения зависит от его типа:

   - **Строка** и **Число** — введите значение вручную.
   - **Дата** — введите значение вручную или задайте с помощью панели выбора даты и времени.
   - **Логическое** — выберите значение из выпадающего списка: **True** или **False**.

Чтобы удалить переменную, нажмите на кнопку в строке ее настроек.

Чтобы использовать указанную переменную в RQL-запросе, необходимо использовать шаблон Handlebars:

- Шаблоны для обращения к переменной типа **Строка**, **Число** или **Логическое**:
  - `{{<variable_argument>}}` — обращение к переменной с экранированием специальных символов в ее значении;
  - `{{{<variable_argument>}}}` — обращение к переменной без экранирования специальных символов в ее значении.
- Шаблоны для обращения к переменной типа **Дата**:
  - `{{sqlDateTime <variable_argument>}}` — обращение к значению типа DateTime64;
  - `{{sqlDateTime <variable_argument> false}}` — обращение к значению типа DateTime.

Здесь:

- `<variable_argument>` — аргумент переменной.

Если аргумент переменной содержит специальные символы или `0`, то при обращении к переменной необходимо экранировать ее аргумент одинарными кавычками, например: `{{'<variable_argument>'}}`.

Пример использования переменных в RQL-запросе

Предположим, что нам необходимо получить события, отфильтрованные по значениям `evo.siem` и `evo.core` поля `moduleKey`.

Чтобы передавать конкретные значения в RQL-запрос, можно создать две переменные:

- Первая переменная:
  - Аргумент: `moduleKey1`.
  - Тип значения: **Строка**.
  - Значение: `evo.siem`.
- Вторая переменная:
  - Аргумент: `moduleKey2`.
  - Тип значения: **Строка**.
  - Значение: `evo.core`.

Тогда для решения поставленной задачи можно использовать следующий запрос:

```
SELECT id, moduleKey, tenantId WHERE moduleKey IN
("{{moduleKey1}}","{{moduleKey2}}") ORDER BY tenantId
```

13. Укажите параметры в секции **Сопоставление полей**.

При использовании группировки на виджетах отображаются только первые 20 групп.

Набор доступных параметров зависит от выбранного типа диаграммы:

- **Столбчатая диаграмма** и **Столбчатая диаграмма с накоплением**: доступны поля **Легенда**, **Значение** и **Группировка** (последнее поле обязательно для типа **Столбчатая диаграмма с группировкой**).

  Столбчатая диаграмма позволяет отображать значения показателей в виде вертикальных столбцов. Высота столбцов пропорциональна значениям.

  Столбчатая диаграмма с накоплением позволяет сравнивать сгруппированные показатели с другими группами тех же показателей, например, распределение DDOS-атак по IP-адресам с их группировкой по типам. Столбцы значений в каждой группе расположены друг поверх друга, формируя единый столбец в рамках группы.

Пример

Необходимо построить диаграмму, показывающую распределение количества IP-адресов DDOS-атак по IP-адресам с их группировкой по типам.

Запрос:

```
SELECT ip, attack_type, (count(id) AS cnt GROUP BY Ip,
attack_type
```

В сопоставлении полей указываются следующие атрибуты:

- **Легенда** — `ip` (IP, замеченные в атаке);
- **Значение** — `cnt` (количество записей);
- **Группировка** — `attack_type` (тип атаки).

◦ **Круговая диаграмма**: доступны поля **Легенда** и **Значение**.

Пример

Необходимо построить диаграмму, показывающую распределение количества атак по типам.

Запрос:

```
SELECT attack_type, sum(involved_attacks) AS total_attacks GROUP
BY attack_type
```

В сопоставлении полей указываются следующие атрибуты:

- **Легенда** — `attack_type` (тип атаки);
- **Значение** — `total_attacks` (суммарное количество атак).

◦ **График**: доступны поля **Ось X**, **Ось Y** и **Группировка**.

Пример

Необходимо построить график, показывающий зависимость количества атак с каждого IP-адреса в зависимости от оценки его репутации.

Запрос:

```
SELECT ip, reputation_score, sum(involved_attacks) AS
total_attacks GROUP BY ip, reputation_score
```

В сопоставлении полей указываются следующие атрибуты:

- **Ось X** — `reputation_score` (оценка рейтинга);
- **Ось Y** — `total_attacks` (суммарное количество атак).

◦ **Итоги**: доступно поле **Значение**.

Пример

Необходимо вывести количество IP, замеченных в DDOS-атаках.

Запрос:

```
SELECT uniq(ip) AS uniq
```

В поле **Значение** указывается `uniq` — псевдоним атрибута, возвращающего количество уникальных значений в столбце `ip`.

- **Таблица**: нужно заполнить для каждого столбца таблицы по два поля: **Название колонки** и **Атрибут из запроса**. Чтобы создать новый столбец в таблице, нажмите на кнопку **Добавить колонку**.

Пример

Необходимо создать таблицу с информацией об IP-адресах из определенной подсети, замеченных в DDOS-атаках. Таблица должна быть отсортирована по времени последнего обнаружения от позднего к раннему.

Запрос:

```
SELECT ip, attack_type, involved_attacks, last_detected WHERE ip
IN '192.0.2.0/24' ORDER BY last_detected DESC
```

Название колонки	Атрибут из запроса
IP	`ip` (атрибут, возвращающий IP источника атаки)
Тип атаки	`attack_type`
Количество атак	`involved_attacks`
Время	`last_detected` (атрибут, возвращающий метку времени последнего обнаружения)

14. Если вы хотите, чтобы данные на виджете регулярно обновлялись, воспользуйтесь переключателем **Включить автообновление** и укажите периодичность обновления:
    a. Задайте частоту обновления вручную или с помощью кнопок — и +.
    b. Выберите из выпадающего списка единицы времени периода.
15. Чтобы предварительно просмотреть, как будут отображаться данные на виджете в соответствии с заданными настройками, нажмите на кнопку **Обновить** (🔄) в правой верхней части окна создания виджета. Доступны два варианта предпросмотра:
    a. **Диаграмма** — отображает визуальное представление виджета на дашборде.
    b. **Данные** — отображает результаты RQL-запроса в табличной форме.
16. Нажмите на кнопку **Добавить виджет**. Система создаст виджет и отобразит соответствующее уведомление. Новый виджет появится на дашборде.

Обновление виджета

Чтобы отобразить актуальные данные на виджете, требуется его обновить. Вы можете принудительно обновить виджет вручную или настроить автоматическое обновление.

Обновление виджета вручную

Чтобы обновить данные виджета вручную:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку ↻ в правом верхнем углу выбранного виджета. Данные на виджете будут обновлены.

Автоматическое обновление виджета

Чтобы настроить регулярное автоматическое обновление виджета:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку действий (⋮) в верхнем правом углу виджета и выберите опцию **Редактировать**. Отобразится окно редактирования виджета.
5. Включите переключатель **Включить автообновление** и укажите периодичность регулярных обновлений:
   a. Задайте частоту обновления вручную или с помощью кнопок — и +.
   b. Выберите из выпадающего списка единицы времени периода.
6. Нажмите на кнопку **Изменить виджет**. Система сохранит измененные данные виджета. Данные на виджете будут автоматически обновляться по истечении указанного периода времени.

Изменение размеров виджета

Чтобы изменить размеры (высоту и ширину) виджета:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Наведите курсор на правый нижний угол выбранного виджета. Курсор примет вид двунаправленной стрелки.
5. Потяните за правый нижний угол виджета:
   - вправо/влево — для изменения ширины;
   - вверх/вниз — для изменения высоты.

   Размеры виджета будут изменены.

Перемещение виджета

Чтобы переместить виджет на дашборде:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Наведите курсор на область названия выбранного виджета. Курсор примет вид четырехнаправленной стрелки.
5. Перетащите виджет в нужное положение на дашборде. Виджет будет перемещен.

Изменение настроек виджета

Чтобы изменить настройки виджета:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку действий (⋮) в верхнем правом углу виджета и выберите опцию **Редактировать**. Отобразится окно редактирования виджета.
5. Внесите изменения в требуемые поля.
6. Нажмите на кнопку **Изменить виджет**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится на дашборде.

Настройка отображения легенды виджета

По умолчанию при добавлении виджета в правой его части отображается легенда, на которой приведены цветовые обозначения и названия рядов данных на виджете.

Легенда не отображается для виджетов событий с типом **Таблица** или **Итоги**.

Чтобы скрыть легенду, нажмите на кнопку действий (⋮) в верхнем правом углу виджета и выберите опцию **Скрыть легенду**. Легенда перестанет отображаться на виджете, а диаграмма визуализации данных займет всю ширину виджета.

Чтобы отобразить скрытую легенду, нажмите на кнопку действий (⋮) в верхнем правом углу виджета и выберите опцию **Показать легенду**. В правой части виджета отобразится его легенда, а диаграмма визуализации отмасштабируется по оставшемуся на виджете пространству.

Настройка отображения рядов данных виджета

По умолчанию при добавлении виджета отображаются все его ряды данных.

Чтобы скрыть ряд данных на виджете, нажмите на его цветовое обозначение в легенде. Ряд перестанет отображаться на диаграмме визуализации данных, а его цветовое обозначение в легенде станет серым.

Чтобы отобразить скрытый ряд данных на виджете, нажмите на его цветовое обозначение в легенде. Ряд отобразится на диаграмме визуализации данных, а его обозначение в легенде вернет свой изначальный цвет.

Просмотр деталей ряда данных виджета

Просмотр деталей ряда данных доступен для виджетов событий.

Просмотр событий ряда данных

Просмотр событий доступен только для виджетов событий следующих типов:

- Столбчатая диаграмма.
- График.
- Круговая диаграмма.
- Столбчатая диаграмма с накоплением.

На виджетах должна быть задана группировка только по одному атрибуту (например: `GROUP BY collectorId`).

Чтобы просмотреть детали событий ряда данных, нажмите на название соответствующего ряда в легенде. Система откроет страницу раздела **Поиск**, на которой отобразится список событий, отфильтрованный в соответствии с выбранным рядом данных.

Создание копии виджета

Чтобы создать копию виджета:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку действий (⋮) в верхнем правом углу виджета и выберите опцию **Создать копию**. Система создаст копию виджета и отобразит соответствующее уведомление. Новый виджет появится на дашборде в конце списка.

Удаление виджета

Чтобы удалить виджет:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку действий (⋮) в верхнем правом углу виджета и выберите опцию **Удалить**. Система отобразит окно подтверждения удаления виджета.
5. Нажмите на кнопку **Удалить**. Система удалит виджет и отобразит соответствующее

уведомление. Удаленный виджет исчезнет с дашборда.

Экспорт виджета

Чтобы экспортировать виджет:

1. Перейдите в раздел **Дашборды**.
2. Нажмите на стрелку вниз в верхней части раздела. Отобразится выпадающий список дашбордов.
3. Выберите требуемый дашборд из списка. Система отобразит виджеты, содержащиеся на дашборде.
4. Нажмите на кнопку **Экспорт** (↑) в правом верхнем углу выбранного виджета. Отобразится окно экспорта виджета.
5. Нажмите на кнопку **Экспортировать**. Система загрузит данные виджета на устройство пользователя в формате CSV и отобразит соответствующее уведомление.

Работа с отчетами

Данный раздел описывает процесс работы с отчетами и шаблонами в системе. Работа осуществляется в разделе **Отчеты** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Об отчетах

Отчеты в системе создаются для сбора, анализа и представления данных, собранных системой.

Пользователи могут создавать, просматривать и управлять отчетами, что включает в себя настройку параметров отчета, его создание на основе предопределенных шаблонов, а также экспорт в файл. Отчеты могут быть настроены для автоматической публикации по расписанию.

Интерфейс раздела

Интерфейс раздела зависит от открытой вкладки. Доступные вкладки:

- **Сформированные** — вкладка для работы со сформированными отчетами.
- **Шаблоны** — вкладка для работы с шаблонами отчетов.

Интерфейс вкладки Сформированные

Панель инструментов вкладки содержит следующие элементы:

- Поле **Поиск** предназначено для быстрого поиска отчетов по их названию.
- Кнопка **Фильтр** (▽) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка отчетов.

В рабочей области вкладки отображается таблица имеющихся отчетов. Таблица содержит

следующие столбцы:

- **Название** — имя отчета, используемое для его идентификации в системе.
- **Теги** — теги, ассоциированные с отчетом.
- **Дата публикации** — дата и время публикации отчета в системе.

При работе с таблицей отчетов доступны следующие операции:

- Поиск отчетов по полю **Название**.
- Фильтрация отчетов по полю **Теги**. Теги вводятся вручную и применяются по клавише ENTER.
- Сортировка отчетов по полям **Название** и **Дата публикации**.
- Настройка отображения таблицы.

При выборе конкретного отчета в правой части рабочей области отображается его карточка с детальной информацией.

Интерфейс вкладки Шаблоны

Панель инструментов вкладки содержит следующие элементы:

- Кнопка **Создать** (+) позволяет создать новый шаблон отчета.
- Кнопка **Создать из схемы** (+) позволяет создать новый шаблон отчета на основе схемы.
- Поле **Поиск** предназначено для быстрого поиска шаблонов по их названию.
- Кнопка **Фильтр** ( ) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка шаблонов.

В рабочей области вкладки отображается таблица имеющихся шаблонов отчетов. Таблица содержит следующие столбцы:

- **Название** — имя шаблона отчета, используемое для его идентификации в системе.
- **Расписание** — индикатор автоматического создания отчетов на основе этого шаблона по расписанию: **Включено** или **Выключено**.
- **Теги** — список связанных с шаблоном тегов, используемых для его поиска и классификации.
- **Дата изменения** — дата и время последнего изменения шаблона отчета.
- **Дата последнего отчета** — дата и время последней публикации отчета на основе текущего шаблона.

При работе с таблицей шаблонов отчетов доступны следующие операции:

- Поиск шаблонов по полю **Название**.
- Фильтрация шаблонов по полям **Теги** и **Расписание**:
  - Теги вводятся вручную и применяются по клавише ENTER.
  - Индикатор расписания выбирается из выпадающего списка.
- Сортировка шаблонов по полям **Название** и **Дата изменения**.
- Настройка отображения таблицы.

При выборе конкретного шаблона отчета в правой части рабочей области отображается его карточка с детальной информацией.

Работа с шаблоном отчета

Шаблоны отчетов являются инструментом для определения структуры и периодичности публикации отчетов. Они позволяют выбирать аналитические данные для включения в отчеты и поддерживают функции автоматического создания отчетов по расписанию.

Доступные операции над шаблоном отчета:

- Создание шаблона отчета
- Просмотр карточки шаблона отчета
- Изменение шаблона
- Создание копии
- Удаление шаблона
- Сохранение шаблона отчета как схемы

Создание шаблона отчета

Создать шаблон отчета можно следующими способами:

- Создать шаблон, добавляя виджеты вручную, или на основе существующего дашборда.
- Использовать предварительно созданную схему.

Создание шаблона отчета вручную

Чтобы создать шаблон отчета вручную:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит список имеющихся шаблонов отчетов.
2. Нажмите на кнопку **Создать** (+) на панели инструментов. Система отобразит структуру шаблона в рабочей области и боковую панель с его текущими настройками.
3. Введите название шаблона отчета.
4. При необходимости введите описание шаблона отчета.
5. При необходимости добавьте теги для классификации и поиска шаблона в системе.
6. При необходимости выберите из выпадающего списка **Шаблон дашборда** дашборд, который определит структуру текущего шаблона отчета.

Пользователю доступны общие дашборды и созданные им персональные дашборды.

7. Добавьте отдельные виджеты в шаблон с помощью кнопки **Добавить виджет**. Если вы выбрали дашборд как основу для шаблона, вы можете удалить ненужные виджеты.

Шаблон отчета должен содержать хотя бы один виджет.

8. Настройте виджеты, как описано в разделе Добавление виджета. Период, за который собраны данные в отчете, определяется настройками виджетов, добавленных в него.

   Изменения настроек не повлияют на исходный дашборд.

   При экспорте отчета система может изменить его внешний вид для лучшего отображения данных.

9. При необходимости задайте аргументы и значения переменных, которые требуется использовать в RQL-запросах виджетов, в секции **Переменные**.

   Чтобы добавить переменную:

   a. Нажмите на кнопку **Добавить**. Появится строка настройки переменной.

   b. Введите аргумент переменной. Аргумент может содержать только латинские буквы (`A–Z`, `a–z`) и цифры (`0–9`).

   c. Выберите тип значения переменной. Доступные типы:

      - **Строка**;
      - **Число**;
      - **Дата**;
      - **Логическое**.

   d. Укажите значение переменной. Способ задания значения зависит от его типа:

      - **Строка** и **Число** — введите значение вручную.
      - **Дата** — введите значение вручную или задайте с помощью панели выбора даты и времени.
      - **Логическое** — выберите значение из выпадающего списка: **True** или **False**.

   Чтобы удалить переменную, нажмите на кнопку в строке ее настроек.

   Чтобы применить указанные переменные и перестроить виджеты с учетом их значений, нажмите на кнопку **Применить**.

10. Задайте период хранения. Выберите единицы измерения и их количество, чтобы определить срок, по истечении которого отчет будет удален из системы.

11. Задайте параметры публикации.

    Для публикации отчета по расписанию переведите переключатель **Расписание** в активное положение и задайте настройки публикации:

    - **Период** — выберите из выпадающего списка период публикации:
      - **Каждый день** — для публикации отчетов ежедневно.
      - **По дням** — для публикации отчетов в конкретные дни недели. При выборе этого периода необходимо указать дни недели для публикации.
    - **Время** — укажите время, в которое должен быть сформирован отчет.

12. Нажмите **Сохранить**. Система создаст шаблон отчета и отобразит соответствующее

уведомление. Новый шаблон появится в разделе **Отчеты** на вкладке **Шаблоны**.

Чтобы закрыть окно создания шаблона и вернуться на вкладку **Шаблоны**, нажмите на кнопку **Назад** в левой верхней части экрана.

Создание шаблона отчета из схемы

Чтобы создать шаблон отчета из схемы:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит список имеющихся шаблонов отчетов.
2. Нажмите на кнопку **Создать из схемы** в левой части панели инструментов. Откроется окно создания шаблона.
3. Выберите схему из выпадающего списка.
4. Сопоставьте переменные виджетов с источниками данных — хранилищем событий или активным списком в зависимости от типа виджетов.
5. Нажмите на кнопку **Продолжить**. Система отобразит структуру шаблона в рабочей области и боковую панель с его текущими настройками.
6. При необходимости внесите изменения в настройки, как при создании шаблона вручную.

Если при настройке вы выберете дашборд в выпадающем списке **Шаблон дашборда**, виджеты текущей схемы будут заменены на виджеты этого дашборда.

7. Нажмите **Сохранить**. Система создаст шаблон отчета и отобразит соответствующее уведомление. Новый шаблон появится в разделе **Отчеты** на вкладке **Шаблоны**.

Чтобы закрыть окно создания шаблона и вернуться на вкладку **Шаблоны**, нажмите на кнопку **Назад** в левой верхней части экрана.

Просмотр карточки шаблона отчета

Чтобы просмотреть карточку шаблона отчета:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит сведения об имеющихся шаблонах отчетов.
2. Нажмите на строку шаблона в списке. Система отобразит в правой части экрана карточку этого шаблона с базовой информацией о нем.

В верхней части карточки отображается название шаблона отчета. Справа от названия расположена кнопка ⋮, по которой открывается выпадающее меню с действиями, доступными для выбранного шаблона.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка содержит сведения о шаблоне отчета, сгруппированные по следующим секциям:

- **Информация** — основная информация о шаблоне отчета:

- **ID** — уникальный идентификатор шаблона отчета.
- **Название** — имя шаблона отчета.
- **Описание** — дополнительная информация о шаблоне.
- **ID тенанта** — уникальный идентификационный код тенанта, к которому относится шаблон.
- **Теги** — список связанных с шаблоном тегов, используемых для его поиска и классификации.
- **Расписание** — индикатор автоматического создания отчетов на основе этого шаблона по расписанию: **Включено** или **Выключено**.
- **Дата последнего отчета** — дата и время создания последнего отчета по текущему шаблону. Поле отображается, если отчет создавался хотя бы один раз.
- **Дата следующего отчета** — дата и время создания следующего отчета по настроенному расписанию. Поле отображается, если настроена публикация отчета по расписанию.
- **Дата создания** — дата и время создания шаблона отчета.
- **Создал** — пользователь, инициировавший создание шаблона отчета.
- **Дата изменения** — дата и время последнего изменения шаблона отчета.
- **Изменил** — пользователь, выполнивший последнее изменение шаблона отчета.

- **Переменные** — список аргументов и значений переменных, заданных в шаблоне. Секция отображается, если задана хотя бы одна переменная.

В нижней части карточки расположена кнопка **Сформировать**, позволяющая создать версию отчета по выбранному шаблону на текущий момент.

Изменение шаблона

Чтобы изменить шаблон отчета:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит сведения об имеющихся шаблонах отчетов.
2. Нажмите на строку шаблона в списке. Система отобразит в правой части экрана карточку этого шаблона с базовой информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек.
4. Внесите изменения в настройки шаблона.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке шаблона и в списке раздела **Отчеты** на вкладке **Шаблоны**.

Название, описание и теги шаблона отчета можно также изменить из его карточки. Для этого:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит сведения об имеющихся шаблонах отчетов.
2. Нажмите на строку шаблона в списке. Система отобразит в правой части экрана карточку этого шаблона с базовой информацией о нем.

3. Нажмите на кнопку рядом с нужным полем шаблона в его карточке. Система отобразит поле ввода значения.
4. Укажите новое значение поля.
5. Нажмите **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке шаблона и в списке раздела **Отчеты** на вкладке **Шаблоны**.

Чтобы закрыть окно изменения шаблона и вернуться к его карточке, нажмите на кнопку **Назад** в левой верхней части экрана.

Создание копии

Чтобы создать копию шаблона:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит сведения об имеющихся шаблонах отчетов.
2. Нажмите на строку шаблона в списке. Система отобразит в правой части экрана карточку этого шаблона с базовой информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Создать копию**. Система отобразит окно настроек шаблона.
4. Внесите требуемые изменения в настройки шаблона.
5. Нажмите на кнопку **Создать**. Система создаст шаблон отчета и отобразит соответствующее уведомление. Новый шаблон появится в разделе **Отчеты** на вкладке **Шаблоны**.

Удаление шаблона

Чтобы удалить шаблон отчета:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит сведения об имеющихся шаблонах отчетов.
2. Нажмите на строку шаблона в списке. Система отобразит в правой части экрана карточку этого шаблона с базовой информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления шаблона.
4. Нажмите на кнопку **Удалить**. Система удалит шаблон отчета и отобразит соответствующее уведомление. Удаленный шаблон будет исключен из списка шаблонов в разделе **Отчеты** на вкладке **Шаблоны** системы.

Сохранение шаблона отчета как схемы

Чтобы сохранить шаблон отчета как схему:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит сведения об имеющихся шаблонах отчетов.
2. Нажмите на строку шаблона в списке. Система отобразит в правой части экрана карточку этого шаблона с базовой информацией о нем.

3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Создать схему**. Отобразится окно создания схемы.
4. По умолчанию название и описание создаваемой схемы идентичны названию и описанию шаблона. При необходимости измените название и описание схемы.
5. Выберите из выпадающего списка каталог экспертизы, в котором будет создана схема.
6. Для виджетов событий и активных списков автоматически генерируются переменные, которые при создании шаблона отчета из схемы нужно связать с источниками данных. При необходимости измените переменные виджетов.

Чтобы при создании шаблона отчета несколько виджетов получали данные из одного источника, задайте для них одинаковые переменные.

7. Нажмите на кнопку **Далее**. Откроется окно создания RObject-конфигурации схемы с заполненными данными.
8. Задайте идентификатор схемы шаблона отчета в поле `id`.
9. При необходимости внесите изменения в поля схемы.
10. Нажмите на кнопку **Опубликовать версию**. Система создаст схему шаблона отчета и отобразит соответствующее уведомление. Новая схема появится в таблице раздела **Экспертиза**.

Чтобы схема стала доступной для создания шаблонов отчета, ее необходимо включить.

Вы можете создать черновик схемы шаблона без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик схемы и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Работа со сформированным отчетом

Все отчеты в системе формируются на основе шаблонов отчетов.

Доступные операции над отчетом:

- Создание отчета
- Просмотр карточки отчета
- Просмотр содержимого отчета
- Экспорт отчета
- Удаление отчета

Создание отчета

Создание отчета начинается с выбора и настройки шаблона. Вы можете добавить различные виджеты и задать параметры отчета. Отчет можно создать вручную или настроить расписание, чтобы отчеты создавались автоматически.

Чтобы создать отчет из существующего шаблона:

1. Перейдите на вкладку **Шаблоны** раздела **Отчеты**. Система отобразит сведения об имеющихся шаблонах отчетов.
2. Нажмите на строку шаблона в списке. Система отобразит в правой части экрана карточку этого шаблона с базовой информацией о нем.
3. Нажмите **Сформировать** в нижней части карточки. Система создаст по шаблону версию отчета на текущий момент и отобразит соответствующее уведомление. Новый отчет появится в разделе **Отчеты** на вкладке **Сформированные**.

Просмотр карточки отчета

Чтобы просмотреть карточку отчета:

1. Перейдите на вкладку **Сформированные** раздела **Отчеты**. Система отобразит сведения об имеющихся отчетах.
2. Нажмите на строку отчета в списке. Система отобразит в правой части экрана карточку этого отчета с базовой информацией о нем.

В верхней части карточки отображается название отчета. Справа от названия расположена кнопка для удаления отчета.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка содержит сведения об отчете, сгруппированные по следующим секциям:

- **Информация** — основная информация об отчете:
  - **ID** — уникальный идентификатор отчета.
  - **Название** — имя отчета, используемое для идентификации в системе.
  - **ID тенанта** — уникальный идентификационный код тенанта, к которому относится отчет.
  - **Теги** — список тегов, связанных с отчетом.
  - **Шаблон** — шаблон, по которому был создан отчет.
  - **Дата публикации** — дата и время публикации отчета.
  - **Срок хранения** — срок хранения отчета, по истечении которого он будет удален из системы.
  - **Сформировал** — имя пользователя, сформировавшего отчет.
  - **Запуск** — каким образом был сформирован отчет: **Автоматически** или **Вручную**.
- **Переменные** — список аргументов и значений переменных, заданных в шаблоне. Секция отображается, если задана хотя бы одна переменная.

В нижней части карточки расположены кнопки:

- **Смотреть отчет** — позволяет просмотреть содержимое отчета.
- **Экспорт в файл** — позволяет экспортировать отчет.

Просмотр содержимого отчета

Чтобы просмотреть содержимое отчета в формате HTML:

1. Перейдите на вкладку **Сформированные** раздела **Отчеты**. Система отобразит сведения об имеющихся отчетах.
2. Нажмите на строку отчета в списке. Система отобразит в правой части экрана карточку этого отчета с базовой информацией о нем.
3. Нажмите на кнопку **Смотреть отчет** в нижней части карточки шаблона. Система отобразит содержимое отчета в отдельной вкладке браузера в формате HTML.

Чтобы закрыть отчет и вернуться к списку отчетов, нажмите на кнопку **Назад** в левой верхней части экрана.

Экспорт отчета

Чтобы выгрузить отчет:

1. Перейдите на вкладку **Сформированные** раздела **Отчеты**. Система отобразит сведения об имеющихся отчетах.
2. Нажмите на строку отчета в списке. Система отобразит в правой части экрана карточку этого отчета с базовой информацией о нем.
3. Нажмите на кнопку **Экспорт в файл** в нижней части карточки. Система отобразит окно экспорта.
4. Выберите формат экспорта из выпадающего списка: HTML, PDF, CSV.

Для формата CSV отчет выгружается в виде ZIP-архива.

5. Нажмите на кнопку **Экспорт**. Система выгрузит отчет на устройство пользователя в выбранном формате и отобразит соответствующее уведомление.

При экспорте отчетов их внешний вид может измениться для удобства отображения данных:

- Все временные метки преобразуются в формат UTC.
- Шаги делений на виджетах могут измениться в зависимости от разрешения экрана для лучшего отображения данных.
- В файлах PDF таблицы могут содержать до восьми столбцов. Если столбцов больше четырех, лист с таблицей разворачивается горизонтально.

Удаление отчета

Чтобы удалить отчет:

1. Перейдите на вкладку **Сформированные** раздела **Отчеты**. Система отобразит сведения об имеющихся отчетах.
2. Нажмите на строку отчета в списке. Система отобразит в правой части экрана карточку этого отчета с базовой информацией о нем.

3. Нажмите на кнопку в верхнем правом углу карточки. Отобразится окно подтверждения удаления отчета.
4. Нажмите на кнопку **Удалить**. Система удалит отчет и отобразит соответствующее уведомление. Удаленный отчет будет исключен из списка раздела **Отчеты** на вкладке **Сформированные**.

Работа с разделом Поиск

Раздел **Поиск** позволяет просматривать и анализировать события и записи активных списков. В этом разделе отображаются события из указанного хранилища или записи указанного активного списка, с возможностью поиска и фильтрации вхождений с помощью пользовательских фильтров и запросов на языке RQL.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О событиях

Событие — это идентифицированный переход системы, сервиса или сети в состояние, которое указывает на возможное нарушение политики информационной безопасности, сбой средств контроля или иную ситуацию, которая может быть значимой для информационной безопасности.

Система позволяет:

- задавать модели событий для сбора данных из нескольких источников;
- собирать события в соответствии с заданными моделями;
- детектировать действия в системе в виде событий аудита;
- обогащать события за счет получения дополнительных данных и установления связей с другими событиями;
- отслеживать и визуализировать события;
- управлять хранением и обработкой событий;
- создавать фиды событий для отправки в R-Vision TIP и иные системы класса Threat Intelligence.

Режимы поиска

В системе доступны два режима поиска:

- **Базовый поиск** — поиск без использования проекций и группировки результатов.
- **Продвинутый поиск** — поиск с использованием проекций и группировки результатов с помощью операторов языка RQL `SELECT` и `GROUP BY` соответственно.

  Пример 33. Пример запроса в режиме продвинутого поиска

  ```
  SELECT count(id) as cnt, collectorId GROUP BY collectorId
  ```

  При выполнении вышеприведенного запроса данные группируются по полю `collectorId` и

вычисляется `cnt` — количество записей в каждой группе. Результат запроса — таблица с двумя столбцами: `cnt` и `collectorId`.

В режиме продвинутого поиска структура результата не определяется моделью события, из-за чего некоторые функции недоступны, а именно:

- график распределения результатов поиска по времени;
- статистика по значениям полей;
- контекстное меню для работы со свойствами в ячейках таблицы;
- настройка списка и порядка полей, отображаемых в таблице результатов поиска;
- группировка полей в карточке события или записи активного списка;
- шаблоны настроек таблицы результатов поиска.

По умолчанию при открытии раздела **Поиск** используется режим базового поиска. Режим поиска для текущего запроса определяется автоматически при нажатии кнопки **Поиск** и отправке запроса:

- Запрос выполняется в режиме продвинутого поиска, если он содержит операторы `GROUP BY`, `RIGHT JOIN`, `FULL JOIN` или `SELECT`, за исключением `SELECT *`.
- В остальных случаях запрос выполняется в режиме базового поиска.

Интерфейс раздела

В верхней части раздела отображается название текущего хранилища или активного списка. Чтобы искать данные в другом хранилище или активном списке, выберите его название из выпадающего списка. Для быстрого поиска хранилища или активного списка в выпадающем списке начните вводить его название.

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка позволяет экспортировать найденные события или записи.
- Кнопка открывает страницу сравнения событий. Кнопка отображается, если для поиска выбрано хранилище событий.
- Кнопка открывает панель для работы с поисковыми шаблонами.
- Кнопка позволяет скопировать в буфер обмена текущий поисковый запрос, объединенный с условиями фильтра оператором `AND`. Скопированный запрос можно использовать при создании виджета.
- Поле запроса предназначено для ввода ключевых слов или RQL-запроса для поиска данных в выбранном хранилище или активном списке.
- Кнопка позволяет выбрать способ задания периода, в рамках которого требуется искать данные.
- Поля периода предназначены для указания временного диапазона поиска. Зависят от выбранного способа задания периода.
- Кнопка **Поиск** позволяет начать поиск данных по заданным параметрам.

При необходимости вы можете остановить процесс поиска, нажав на кнопку

Стоп. В разделе будут отображены только записи, найденные до прерывания поиска.

Если активен режим базового поиска, под панелью отображается график распределения найденных записей по времени. В правом верхнем углу графика находится кнопка для сворачивания и разворачивания.

Вид рабочей области раздела зависит от открытой вкладки. Доступные вкладки:

- **Данные** — содержит информацию о найденных событиях или записях активного списка.
- **Статистика** — отображает сводные данные по полям найденных событий и записей активного списка с возможностью просмотра рейтинга значений по выбранному полю.

Вкладка доступна, если активен режим базового поиска.

Интерфейс вкладки Данные

На вкладке **Данные** отображается список событий или записей. Список представлен в виде таблицы, набор столбцов которой формируется в зависимости от текущего режима поиска:

- в режиме базового поиска — согласно полям модели события выбранного хранилища, для которых задано отображение;
- в режиме продвинутого поиска — согласно полям, заданным с помощью операторов `SELECT` и/или `GROUP BY`.

По умолчанию в таблице на вкладке **Данные** отображаются столбцы, соответствующие следующим полям:

- для универсальной модели события: `timestamp`, `dvendor`, `dproduct`, `name`, `msg`, `dvc`, `outcome`;
- для модели события аудита: `timestamp`, `raw`, `suser`, `eventType`, `moduleKey`, `eventRiskLevel`, `data`, `meta`;
- для пользовательской модели события: `id`, `type`, `timestamp`, `sourceIp`.

При работе с таблицей записей на вкладке **Данные** доступны следующие операции:

- Сортировка по любому столбцу.
- Обновление таблицы.
- Настройка отображения таблицы.

По умолчанию в таблице доступен просмотр не более 500 первых записей.

При выборе конкретной записи в правой части рабочей области отображается ее карточка с детальной информацией.

Интерфейс вкладки Статистика

- Вкладка **Статистика** доступна только в режиме базового поиска.

- Поля с типом данных `array` и `KeyValue` не отображаются в статистике.

На вкладке **Статистика** представлена сводная информация по полям найденных событий или записей активного списка. На вкладке можно просматривать рейтинг значений по каждому из полей, а также общие статистические данные.

Данные по статистике представлены в виде таблицы со следующими столбцами:

- **Поле** — название поля события или записи активного списка.
- **Тип** — тип данных поля.
- **Заполненных значений** — количество заполненных (непустых) значений поля и их процентная доля от общего количества значений.
- **Уникальных значений** — количество уникальных значений поля.
- **Выборка** — количество значений поля, используемых при расчете статистики, и их процентная доля от общего количества.

Поля, которые отсутствуют в хранилище событий, но при этом имеются в универсальной модели, не отображаются в таблице статистики для этого хранилища.

Слева от строки каждого поля в таблице расположена кнопка в виде стрелки, позволяющая показать детальную статистику по выбранному полю.

Вы также можете отобразить статистику сразу всех полей. Для этого необходимо использовать кнопку в виде стрелки слева от области заголовка таблицы.

При работе с таблицей полей на вкладке **Статистика** доступны следующие операции:

- Сортировка по любому столбцу.
- Обновление таблицы.
- Настройка отображения таблицы.

Работа с поиском

Доступные операции при работе с поиском:

- Поиск событий и записей
- Работа с поисковыми шаблонами
- Работа с графиком распределения событий

Поиск событий и записей

Чтобы выполнить поиск по событиям хранилища или записям активного списка:

1. Перейдите в раздел **Поиск**.
2. Выберите хранилище событий или активный список из выпадающего списка в верхней части раздела.

Для быстрого поиска хранилища или активного списка в выпадающем списке начните вводить его название.

3. Задайте критерии поиска:

   a. Чтобы искать записи по ключевым словам или RQL-запросу, введите запрос в поле поиска.

   - Если при поиске находится больше записей, чем указано в настройках отображения, один и тот же RQL-запрос может приводить к разным результатам. Это связано с тем, что по умолчанию поиск записей в хранилище выполняется в произвольном порядке.

     В таких случаях для корректного поиска записей необходимо добавить атрибут `ORDER BY` и явно указать сортировку по требуемому полю.

     Например, для корректного отображения данных с сортировкой по времени требуется явно указать сортировку по полю `timestamp` в RQL-запросе:

     - `ORDER BY timestamp ASC` или `ORDER BY timestamp` — сортировка записей от старых к новым.
     - `ORDER BY timestamp DESC` — сортировка записей от новых к старым.
   - При установке курсора в поле поиска отображается панель с 5 последними запросами.
   - Если в запросе заданы условия поиска по определенным значениям полей с помощью операторов `=` или `IN`, соответствующие поля для найденных записей на вкладке **Данные** будут подсвечиваться.

Если для выбранного хранилища событий и ролей текущего пользователя заданы политики управления доступа к событиям, их правила будут неявно добавлены в секцию `WHERE` поискового запроса.

   b. Чтобы искать записи за период времени, нажмите на кнопку и выберите из выпадающего списка способ задания периода:

      i. **Последний период** — задайте значение вручную или с помощью кнопок — и +, и выберите единицы времени из выпадающего списка. Поиск данных будет выполняться в указанном диапазоне вплоть до нынешнего момента.

      ii. **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск данных.

      iii. **Часто используемые** — выберите в разделе один из предлагаемых периодов.

В нижней части выпадающего списка фильтров расположена секция **История поиска** с 5 последними запросами.

4. Нажмите на кнопку **Поиск**. Список записей, соответствующий заданным критериям, отобразится в таблице. Также над таблицей отобразится количество найденных совпадений.

При необходимости вы можете остановить процесс поиска, нажав на кнопку **Стоп**. В разделе будут отображены только записи, найденные до прерывания поиска.

История запросов поиска хранится в разделе **Диспетчер запросов**.

Поиск по подсетям

Для поиска данных по подсетям используйте оператор `IN`. Формат записи: `%Сетевая маска%/%Префикс%`.

Пример 34. Поиск по подсетям

```
sourceIp IN '::ffff:192.168.1.1/120'
sourceIp IN '2001:db8::/32'
deviceTranslatedAddress IN '203.0.113.0/24'
```

Работа с поисковыми шаблонами

Для удобства поиска можно сохранить текущий запрос в виде шаблона.

Доступные операции над поисковыми шаблонами:

- Создание шаблона
- Поиск шаблона
- Применение шаблона
- Обновление шаблона
- Просмотр конфигурации шаблона

Создание шаблона

Чтобы создать шаблон:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Нажмите на кнопку **Создать шаблон**. Отобразится окно сохранения шаблона.
4. Введите название шаблона.
5. При необходимости введите описание шаблона.
6. Выберите каталог раздела **Экспертиза**, в который будет помещен шаблон.
7. Чтобы сохранить в шаблоне текущий заданный период, установите флажок **Сохранить временной период**.
8. Чтобы сохранить в шаблоне имеющиеся пользовательские фильтры, установите флажок **Сохранить фильтры**.
9. Нажмите на кнопку **Создать**. Система создаст шаблон и отобразит соответствующее уведомление. Новый шаблон появится на панели шаблонов.

Поиск шаблона

Чтобы найти нужный шаблон:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Введите название требуемого шаблона в поле поиска. На панели отобразятся шаблоны, в названии которых встречается введенный запрос.

Применение шаблона

Чтобы применить шаблон:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Выберите шаблон из списка. Система применит шаблон для поиска событий или записей активных списков.

Обновление шаблона

Чтобы обновить существующий шаблон:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Нажмите на кнопку ⇄ в правой части строки нужного шаблона. Отобразится окно обновления шаблона.
4. Внесите необходимые изменения в настройки шаблона.
5. Нажмите на кнопку **Обновить** и подтвердите обновление. Шаблон будет обновлен. В нем будет сохранен поисковый запрос, введенный в поле запроса на момент редактирования. Фильтры и временной период будут сохранены, если установлены флажки **Сохранить временной период** и **Сохранить фильтры**.

Просмотр конфигурации шаблона

Чтобы просмотреть RObject-конфигурацию шаблона:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Нажмите на кнопку ↗ в правой части строки нужного шаблона. Система откроет в новой вкладке браузера раздел **Экспертиза** и отобразит окно просмотра RObject-конфигурации выбранного шаблона поиска.

Работа с графиком распределения событий

График распределения событий доступен только в режиме базового поиска.

График распределения найденных событий или записей по времени отображается после выполнения поискового запроса.

Результаты на графике расположены от более старых к более новым. Для построения графика используются следующие временные значения:

- При поиске в хранилище событий — временные метки событий (`timestamp`).
- При поиске в активных списках — дата создания записей.

Результаты разбиваются на периоды, длительность которых зависит от общего периода поиска. При наведении курсора на определенный период отображаются даты и время его начала и окончания, а также количество событий или записей в этом периоде.

Доступные операции над графиком:

- Отображение и скрытие графика
- Уточнение периода поиска с помощью графика

Отображение и скрытие графика

Панель с графиком по умолчанию свернута. Чтобы отобразить график, нажмите на кнопку **Показать график** (⌄). Над таблицей найденных записей раскроется панель с графиком распределения по времени.

Чтобы скрыть график, нажмите на кнопку **Скрыть график** (⌃). Панель с графиком распределения найденных событий по времени свернется.

Уточнение периода поиска с помощью графика

Вы можете уточнить временной период поискового запроса с помощью графика. Для этого:

1. Наведите курсор на график. Курсор примет форму крестика.
2. Нажмите на нужный период или, удерживая курсор, выделите несколько периодов. Система скорректирует поисковый запрос с учетом выбранного временного диапазона и отобразит уточненные данные в таблице и на графике.

Работа со списком событий или записей активного списка

Список найденных событий или записей отображается в виде таблицы на вкладке **Данные**.

Доступные операции над списком событий или записей:

- Настройка количества отображаемых событий
- Пользовательская фильтрация списка событий
- Автоматическая фильтрация списка по значению поля
- Настройка таблицы

Настройка количества отображаемых событий

По умолчанию отображаются не более 500 первых найденных записей.

Чтобы изменить количество отображаемых записей, выберите значение из выпадающего списка в нижней части страницы. Доступные варианты: 500, 1000, 5000, 10 000, 25 000.

Пользовательская фильтрация списка событий

Доступные операции над пользовательскими фильтрами:

- Добавление фильтра
- Применение фильтра
- Изменение фильтра
- Отключение фильтра
- Включение фильтра
- Инвертирование фильтра
- Удаление фильтра

Если в секции `WHERE` RQL-запроса заданы условия фильтрации, при формировании запроса в БД они объединяются оператором `AND` с фильтрами, настроенными через графический интерфейс.

Добавление фильтра

Чтобы добавить пользовательский фильтр списка:

1. Нажмите на кнопку **Добавить фильтр**. Отобразится окно добавления фильтра.
2. Выберите из выпадающего списка поле, по которому будут фильтроваться записи.

Выбор полей ограничен моделью выбранного хранилища событий или активного списка. Условия фильтрации полей, не входящих в модель (например, для псевдонимов), в режиме продвинутого поиска можно задать в секции `WHERE` RQL-запроса.

3. Выберите из выпадающего списка оператор, который будет применен к значению поля:
   - `=` — равно.

Если в фильтре задано условие поиска по определенному значению поля с помощью оператора `=`, соответствующее поле для найденных записей на вкладке **Данные** будет подсвечиваться.

   - `!=` — не равно.
   - `In` — совпадает с перечисленными значениями. Значения указываются через запятую.

Если в фильтре задано условие поиска по определенному значению поля с помощью оператора `In`, соответствующее поле для найденных записей на вкладке **Данные** будет подсвечиваться.

   - `Like` — соответствует заданному шаблону.

Шаблон может включать обычные символы и следующие метасимволы:

- `%` — обозначает любое количество произвольных символов или отсутствие символов.
- `_` — обозначает один произвольный символ.
- `\` — используется для экранирования символов `%`, `_` и `\`.

  - `>` — больше.
  - `<` — меньше.
  - `>=` — больше или равно.
  - `<=` — меньше или равно.
  - `Between` — находится в заданном диапазоне.

4. Укажите значение поля.
5. Если нужно инвертировать фильтр, включите переключатель **Инвертировать (NOT)**.
6. Если нужно указать название фильтра, включите переключатель **Настроить заголовок** и введите название.
7. Нажмите на кнопку **Добавить**. Созданный фильтр появится рядом с кнопкой **Добавить фильтр**.

Если вы не указывали название фильтра, система создаст его автоматически на основе данных о выбранном поле, операторе и значении.

Если задан хотя бы один фильтр, система отобразит кнопки:

- **Сбросить все** — удаление всех имеющихся фильтров.
- — выпадающее меню команд фильтрации:
  - **Отключить все** — отключение всех фильтров.
  - **Включить все** — включение всех фильтров.
  - **Инвертировать (NOT) все** — инвертирование всех фильтров.
  - **Удалить все** — удаление всех фильтров.

Чтобы уточнить период поиска, используйте график распределения по времени.

Применение фильтра

Чтобы применить настроенный фильтр списка, нажмите на кнопку **Поиск**. Будут найдены события, удовлетворяющие условиям фильтра.

Изменение фильтра

Чтобы изменить заданный пользовательский фильтр списка:

1. Нажмите на стрелку слева от названия фильтра. Отобразится выпадающее меню действий над фильтром.
2. Выберите опцию **Изменить** в выпадающем меню. Откроется окно изменения фильтра.
3. Внесите изменения в требуемые поля.

4. Нажмите на кнопку **Сохранить**. Измененные настройки фильтра будут сохранены.

Отключение фильтра

По умолчанию пользовательский фильтр создается включенным и учитывается при фильтрации списка.

Чтобы отключить фильтр:

1. Нажмите на стрелку слева от названия фильтра. Отобразится выпадающее меню действий над фильтром.
2. Выберите опцию **Отключить** в выпадающем меню. Фильтр будет отключен и не будет использоваться для фильтрации списка событий.

Вы также можете отключить сразу все настроенные фильтры. Для этого выберите опцию **Отключить все** в выпадающем меню кнопки .

Включение фильтра

Чтобы включить пользовательский фильтр:

1. Нажмите на стрелку слева от названия фильтра. Отобразится выпадающее меню действий над фильтром.
2. Выберите опцию **Включить** в выпадающем меню. Фильтр будет применен к списку событий.

Вы также можете включить сразу все настроенные фильтры. Для этого выберите опцию **Включить все** в выпадающем меню кнопки .

Инвертирование фильтра

Чтобы инвертировать условие пользовательского фильтра:

1. Нажмите на стрелку слева от названия фильтра. Отобразится выпадающее меню действий над фильтром.
2. Выберите опцию **Инвертировать (NOT)** в выпадающем меню. Условие фильтра будет заменено на противоположное.

Вы также можете инвертировать условия сразу всех настроенных фильтров. Для этого выберите опцию **Инвертировать (NOT) все** в выпадающем меню кнопки .

Удаление фильтра

Чтобы удалить пользовательский фильтр:

1. Нажмите на стрелку слева от названия фильтра. Отобразится выпадающее меню действий над фильтром.
2. Выберите опцию **Удалить** в выпадающем меню. Фильтр будет сброшен без возможности восстановления.

Вы также можете удалить сразу все настроенные фильтры. Для этого нажмите на кнопку **Сбросить все** или выберите опцию **Удалить все** в выпадающем меню кнопки .

Автоматическая фильтрация списка по значению поля

Автоматическая фильтрация доступна только в режиме базового поиска.

Вы можете настроить автоматическую фильтрацию списка несколькими способами:

- Настройка автоматической фильтрации в списке
- Настройка автоматической фильтрации в карточке
- Настройка автоматической фильтрации по статистике поля

Настройка автоматической фильтрации в списке

Чтобы настроить автоматическую фильтрацию списка по значению поля:

1. Наведите курсор на значение требуемого поля какого-либо события или записи в списке. Рядом с полем отобразятся кнопки фильтрации:
   - — искать события или записи с этим значением.

     Если применить данный фильтр по значению поля, соответствующие поля для найденных записей на вкладке **Данные** будут подсвечиваться.

   - — исключить события или записи с этим значением. Исключение задается через автоматического создание инвертированного фильтра по выбранному значения поля.
2. Нажмите на одну из кнопок фильтрации для автоматического создания фильтра. Заданный фильтр отобразится в списке фильтров слева от кнопки **Добавить фильтр**.

Настройка автоматической фильтрации в карточке

Чтобы настроить автоматическую фильтрацию списка по значению поля из карточки события:

1. Нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.
2. Наведите курсор на требуемое поле в карточке. Система отобразит кнопки фильтрации в строке этого поля:
   - — искать события или записи с этим значением.

     Если применить данный фильтр по значению поля, соответствующие поля для найденных записей на вкладке **Данные** будут подсвечиваться.

   - — исключить события или записи с этим значением. Исключение задается через автоматического создание инвертированного фильтра по выбранному значения поля.
3. Нажмите на одну из кнопок фильтрации для автоматического создания фильтра. Заданный фильтр отобразится в списке фильтров слева от кнопки **Добавить фильтр**.

Настройка автоматической фильтрации по статистике поля

На панели статистики поля доступна фильтрация списка по значениям выбранного поля, а также по информации о датах его значений.

Чтобы настроить автоматическую фильтрацию списка по данным статистики поля:

1. Нажмите на кнопку действий (⋮) в заголовке столбца нужного поля и выберите опцию **Статистика**. В правой части рабочей области отобразится панель с детальной статистикой по столбцу выбранного поля.
2. Наведите курсор на строку требуемого поля на панели статистики. Система отобразит кнопки фильтрации в строке этого поля:
   - ⊞ — искать события или записи с этим значением.

     Если применить данный фильтр по значению поля, соответствующие поля для найденных записей на вкладке **Данные** будут подсвечиваться.

   - ⊟ — исключить события или записи с этим значением. Исключение задается через автоматического создание инвертированного фильтра по выбранному значения поля.
3. Нажмите на одну из кнопок фильтрации для автоматического создания фильтра. Заданный фильтр отобразится в списке фильтров слева от кнопки **Добавить фильтр**.

Настройка таблицы

Доступные операции по настройке таблицы:

- Перемещение столбцов таблицы
- Выбор отображаемых полей таблицы
- Работа с шаблонами настроек таблицы

- На панели настроек отображаются шаблоны, применяемые к таблице текущей вкладки.
- Шаблоны для вкладок **Данные** и **Статистика** настраиваются и сохраняются отдельно друг от друга.
- Каждый набор шаблонов привязан к конкретному хранилищу событий или активному списку.

Перемещение столбцов таблицы

Чтобы переместить столбец, на вкладке **Данные** нажмите на кнопку действий (⋮) в его заголовке и выберите направление перемещения:

- **Переместить влево** (←) — меняет местами выбранный столбец и столбец слева.

  Данная опция недоступна для первого столбца в таблице.

- **Переместить влево** (→) — меняет местами выбранный столбец и столбец справа.

Данная опция недоступна для последнего столбца в таблице.

Выбор отображаемых полей таблицы

Настройка отображаемых полей таблицы доступна только в режиме базового поиска. В режиме продвинутого поиска отображаемые поля определяются выполняемым RQL-запросом.

Чтобы выбрать отображаемые поля таблицы, используйте панель настроек либо карточку события или записи.

Чтобы настроить отображение полей с помощью карточки события или записи:

1. На вкладке **Данные** нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.
2. Наведите курсор на требуемое поле в карточке. В строке поля отобразятся кнопки действий над ним.
3. Нажмите на кнопку ⋮ и задайте параметры отображения в выпадающем меню:
   - **Скрыть колонку** — скрытие столбца поля из таблицы;
   - **Отобразить колонку** — отображение столбца поля в таблице.

Вы также можете скрыть ненужные поля напрямую из таблицы. Для этого на вкладке **Данные** нажмите на кнопку действий (⋮) в заголовке нужного столбца и выберите опцию **Скрыть колонку** ( ).

Если не выбрано ни одно поле для отображения, в таблице отображаются следующие поля:

- `timestamp` — дата и время получения события коллектором.
- `summary` — все заполненные поля найденного события в формате `<field> : <value>`.

  Здесь:

  - `<field>` — название поля;
  - `<value>` — значение поля.

Работа с шаблонами настроек таблицы

Шаблоны настроек таблицы доступны только в режиме базового поиска.

Для удобства можно сохранить часто используемые настройки таблицы в виде шаблона.

Доступные операции над шаблонами настроек таблицы:

- Создание шаблона настроек таблицы
- Применение шаблона настроек таблицы

- Поиск шаблона настроек таблицы
- Изменение шаблона настроек таблицы
- Удаление шаблона настроек таблицы

Создание шаблона настроек таблицы

Чтобы создать шаблон настроек таблицы:

1. Нажмите на кнопку **Сохранить как шаблон** в нижней части панели настроек. Отобразится окно сохранения шаблона настроек.
2. Введите название шаблона настроек.
3. Если необходимо сделать шаблон доступным для всех пользователей системы, установите флажок **Сохранить для всех пользователей**.
4. Нажмите на кнопку **Сохранить**. Система сохранит шаблон и отобразит соответствующее уведомление. Новый шаблон появится в выпадающем списке **Шаблон**.

Применение шаблона настроек таблицы

Чтобы применить шаблон настроек таблицы:

1. Выберите из выпадающего списка **Шаблон** требуемый шаблон. Настройки выбранного шаблона будут применены к таблице событий.

Поиск шаблона настроек таблицы

Чтобы найти шаблон настроек таблицы:

1. Введите название требуемого шаблона в поле **Шаблон**. В выпадающем списке отобразятся шаблоны, в названии которых встречается введенный запрос.

Изменение шаблона настроек таблицы

Изменение шаблона настроек происходит путем замены его полей на текущие настройки таблицы.

Чтобы изменить шаблон:

1. Выберите из выпадающего списка **Шаблон** требуемый шаблон.
2. Внесите необходимые изменения в настройки.
3. В выпадающем списке **Шаблон** нажмите на кнопку **Заменить шаблон** (⇄) в строке нужного шаблона. Отобразится окно подтверждения замены настроек шаблона.
4. Нажмите на кнопку **Заменить**. Система сохранит внесенные изменения в шаблоне и отобразит соответствующее уведомление. Настройки шаблона будут изменены.

Удаление шаблона настроек таблицы

Чтобы удалить шаблон настроек таблицы:

1. В выпадающем списке **Шаблон** нажмите на кнопку **Удалить шаблон** (🗑) в строке нужного шаблона. Отобразится окно подтверждения удаления.

2. Нажмите на кнопку **Удалить**. Система удалит шаблон и отобразит соответствующее уведомление. Удаленный шаблон исчезнет из списка.

Просмотр статистики полей

Статистика полей доступна только в режиме базового поиска.

Поля с типом данных `array` и `KeyValue` не отображаются в статистике.

На вкладке **Статистика** представлена сводная статистика по полям всех найденных событий или записей активного списка. Здесь можно просматривать рейтинг значений по полю, а также детализированные статистические данные, включая тип данных, количество значений, процент заполненных и уникальных значений.

Поля, которые отсутствуют в хранилище событий, но при этом имеются в универсальной модели, не отображаются в таблице статистики для этого хранилища.

Таблица на вкладке **Статистика** содержит следующие столбцы:

- **Поле** — название поля события или записи активного списка.
- **Тип** — тип данных поля.
- **Заполненных значений** — отображаются как процент вхождений, содержащих непустые значения, так и их абсолютное количество.
- **Уникальных значений** — количество уникальных значений поля.
- **Выборка** — процент уникальных значений от общего количества, а также их абсолютное количество.

Просмотр детальной статистики по полю

Чтобы отобразить детальную статистику по полю:

1. Перейдите на вкладку **Статистика**.
2. Нажмите на стрелку ⌄ в строке нужного поля. В раскрывшейся панели система отобразит детализированную статистику, сгруппированную по следующим секциям:
    - **Общие сведения**:
        - **Тип поля** — тип данных поля.
        - **Заполненных значений** — процентная доля заполненных значений (не пустых) от общего количества значений поля.
        - **Уникальных значений** — количество уникальных (неповторяющихся) значений поля.
        - **Выборка** — процентное соотношение количества уникальных значений к общему числу значений поля.
    - **Рейтинг значений**:
        - Частота встречаемости значения, указана в процентах.
        - Количество повторений значения в событиях или записях, приведено в скобках.

- Кнопки для фильтрации событий по значению и копирования значения в буфер обмена.

В рейтинге отображаются до 10 самых частых или редких значений, в зависимости от направления сортировки.

При необходимости вы можете загрузить следующие 10 значений при их наличии, нажав на кнопку **Загрузить еще**.

- **Даты**:

Секция **Даты** отображается для полей с типом данных `DateTime` вместо секции **Рейтинг значений**.

- **Первая дата** — минимальное значение даты из всех найденных событий.
- **Последняя дата** — максимальное значение даты из всех найденных событий.

Детальную статистику по полю можно также просмотреть на вкладке **Данные**:

- Через меню действий:
  1. Наведите курсор на какое-либо значение требуемого поля в списке. Рядом с полем отобразятся кнопки действий.
  2. Нажмите на кнопку действий (⋮) в заголовке столбца нужного поля и выберите опцию **Статистика**. В правой части рабочей области отобразится панель с детальной статистикой по столбцу выбранного поля.
- Через карточку события или записи:
  1. Нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.
  2. Наведите курсор на требуемое поле в карточке. В строке поля отобразятся кнопки действий.
  3. Нажмите на кнопку ⋮ и выберите опцию **Статистика**. Вместо карточки события или записи в правой части рабочей области отобразится панель с детальной статистикой по столбцу выбранного поля.

При наведении курсора на строку любого поля на панели статистики, в строке отображаются следующие кнопки:

- — кнопка копирования данных из этого поля в буфер обмена.
- и — кнопки фильтрации списка событий по данным из этого поля.

Работа с событием или записью активного списка

Доступные операции над событиями или записями активного списка:

- Просмотр события или записи
- Копирование события или записи
- Копирование данных поля

- Экспорт событий и записей

Просмотр события или записи

Чтобы просмотреть найденное событие или запись активного списка:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.

В верхней части карточки отображается название события или записи. Справа от названия расположены кнопки для добавления события в сравнение и копирования события или записи.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

В режиме базового поиска поля карточки события или записи активного списка сгруппированы следующим образом:

- Карточка события содержит следующие поля:
  - **Дата и время** — поля **art** и **timestamp** с датой и временем поступления в коллектор.
  - **Служебные данные** — все служебные поля модели события, кроме полей **raw**, **originalTimestamp**, **timestamp**.
  - **Сведения о событии** — все регулярные поля модели события.
  - (Для базового события) **Исходное событие** — исходное событие, лежащее в основе просматриваемого события.
  - (Для события корреляции) **Базовые события** — список базовых событий с полями **ID** и **Timestamp**.
- Карточка записи активного списка содержит группы полей:
  - **Дата и время** — время создания и обновления записи.
  - **Поля активного списка** — поля согласно схеме активного списка.

В режиме продвинутого поиска поля карточки не сгруппированы и отображаются в порядке, указанном в секции `SELECT` RQL-запроса.

При наведении курсора на строку любого поля отображаются следующие кнопки (только в режиме базового поиска):

- ⊞ и ⊟ — кнопки фильтрации списка по данным из этого поля;
- ⋮ — кнопка, открывающая меню следующих дополнительных опций над полем:
  - копирование данных из этого поля в буфер обмена;
  - скрытие и отображение столбца этого поля в списке;
  - просмотр статистики столбца этого поля.

Копирование события или записи

Чтобы скопировать информацию в буфер обмена:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.
3. Нажмите на кнопку в верхней части карточки. Система сохранит данные в буфер обмена и отобразит соответствующее уведомление.

Копирование данных поля

Вы можете копировать данные полей непосредственно из списка или из карточки. Статистику по конкретному полю можно скопировать из панели статистики соответствующего поля.

Копирование данных из списка

Чтобы скопировать данные поля из списка:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Наведите курсор на значение требуемого поля какого-либо события или записи в списке. Рядом с полем отобразятся кнопки действий над ним.
3. Нажмите на кнопку ⋮ и выберите опцию **Копировать**. Данные выбранного поля будут скопированы в буфер обмена.

Копирование данных из карточки

Чтобы скопировать данные поля из карточки:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.
3. Наведите курсор на требуемое поле в карточке. В строке поля отобразятся кнопки действий над ним.
4. Нажмите на кнопку ⋮ и выберите опцию **Копировать**. Данные выбранного поля будут скопированы в буфер обмена.

Копирование данных статистики поля

В режиме базового поиска на панели статистики доступно копирование значений выбранного поля, а также временных меток соответствующих значений.

Чтобы скопировать данные с панели статистики поля:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.

2. Нажмите на кнопку действия (⋮) в заголовке столбца нужного поля и выберите опцию **Статистика**. В правой части рабочей области отобразится панель с детальной статистикой по столбцу выбранного поля.
3. Наведите курсор на строку требуемого поля на панели статистики. Система отобразит кнопку копирования данных (⧉) в строке этого поля.
4. Нажмите на кнопку копирования данных (⧉). Данные выбранной строки будут скопированы в буфер обмена.

Экспорт событий и записей

При экспорте найденных данных система автоматически выполняет повторный запрос на поиск с теми же параметрами, даже если поиск уже был выполнен вручную. Это может привести к ситуациям, когда в экспорт попадают новые данные, найденные за промежуток времени между ручным поиском и экспортом.

Чтобы избежать таких ситуаций и добиться совпадения экспортированных данных и данных в таблице раздела, необходимо указывать конкретный временной период поиска. Например, с `2025-05-15, 12:00:00` по `2025-05-15, 12:05:00`.

Чтобы экспортировать группу событий или записей активного списка:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на кнопку ⤒ на панели инструментов. Система отобразит окно экспорта.
3. Выберите из выпадающего списка формат экспорта: CSV или JSON.
4. Если установлен флажок **Исключить пустые поля**, из результатов экспорта исключаются поля без значений. Для включения всех полей в результат экспорта снимите этот флажок.
5. Нажмите на кнопку **Экспортировать**. Система загрузит все вхождения, представленные в текущей выдаче, на устройство пользователя одним файлом в выбранном формате и отобразит соответствующее уведомление.

Сравнение событий

Сравнение различных событий между собой позволяет проводить более глубокий анализ и исследование поступающих событий с целью выявления подозрительных отклонений.

Доступные операции по сравнению событий:

- Добавление события в сравнение
- Просмотр сравнения
- Настройка сравнения полей событий
- Копирование событий в сравнении
- Удаление события из сравнения

Добавление события в сравнение

Чтобы добавить событие в сравнение:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.
3. Нажмите на кнопку в верхней части карточки события. Кнопка станет активной, система добавит выбранное событие в сравнение и отобразит соответствующее уведомление.

В сравнение можно добавлять не более десяти событий.

Просмотр сравнения

Чтобы просмотреть добавленные в сравнение события:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на кнопку на панели инструментов раздела. Система отобразит страницу сравнения событий.

Кнопка доступна только при наличии событий, добавленных в сравнение. Вы также можете перейти на страницу просмотра сравнения с помощью кнопки **Перейти в список сравнения**, отображаемой в уведомлении при добавлении события в сравнение.

Чтобы вернуться в раздел **Поиск**, нажмите на кнопку **Назад** в левой верхней части экрана.

Панель инструментов страницы сравнения событий включает в себя следующие компоненты:

- Кнопка копирует данные всех событий в сравнении в буфер обмена.
- Кнопка удаляет все события из сравнения.
- Выпадающий список **Поля** предназначен для выбора конкретных полей событий для сравнения.
- Переключатель **Показать отличия** позволяет отобразить только те поля, значения которых отличаются в каком-либо из сравниваемых событий.
- Переключатель **Скрыть пустые поля** позволяет скрыть поля, не имеющие значений ни в одном из сравниваемых событий.

В рабочей области расположена таблица сравнения событий. Первый столбец таблицы содержит ключи полей событий. В остальных столбцах отображаются данные этих полей для каждого из сравниваемых событий.

В заголовке столбца каждого события расположена кнопка **Удалить из сравнения** () для удаления события из сравнения.

Настройка сравнения полей событий

Чтобы настроить сравнение полей событий:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на кнопку на панели инструментов раздела. Система отобразит страницу сравнения событий.

   Кнопка доступна только при наличии событий, добавленных в сравнение.

3. Выберите из выпадающего списка **Поля** на панели инструментов поля событий, которые требуется сравнить. Если поля не выбраны, сравнение проводится для всех полей событий.
4. Задайте настройки отображения полей событий с помощью переключателей на панели инструментов:
   - **Показать отличия** — включите опцию, если требуется отобразить только те поля событий, в которых имеются отличия.
   - **Скрыть пустые поля** — включите опцию, если требуется скрыть все поля событий, не имеющих значений ни в одном из событий.

Копирование событий в сравнении

Чтобы скопировать данные всех сравниваемых событий в буфер обмена:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на кнопку на панели инструментов раздела. Система отобразит страницу сравнения событий.

   Кнопка доступна только при наличии событий, добавленных в сравнение.

3. Нажмите на кнопку на панели инструментов страницы сравнения событий. Система сохранит данные событий из сравнения в буфер обмена и отобразит соответствующее уведомление.

Удаление события из сравнения

Чтобы удалить событие из сравнения:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на кнопку на панели инструментов раздела. Система отобразит страницу сравнения событий.

   Кнопка доступна только при наличии событий, добавленных в сравнение.

3. Нажмите на кнопку в заголовке нужного события. Система удалит выбранное событие из сравнения и отобразит соответствующее уведомление. Событие перестанет отображаться на странице сравнения.

Вы также можете удалить событие из сравнения из его карточки:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на строку события или записи в списке. Система отобразит в правой части экрана карточку с подробной информацией.
3. Нажмите на кнопку в верхней части карточки события. Кнопка станет неактивной, система удалит выбранное событие из сравнения и отобразит соответствующее уведомление. Событие перестанет отображаться на странице сравнения.

Чтобы удалить сразу все события из сравнения:

1. Перейдите на вкладку **Данные** в разделе **Поиск**. Система отобразит сведения о найденных событиях или записях.
2. Нажмите на кнопку на панели инструментов раздела. Система отобразит страницу сравнения событий.

Кнопка доступна только при наличии событий, добавленных в сравнение.

3. Нажмите на кнопку на панели инструментов страницы сравнения событий. Система отобразит окно подтверждения действия.
4. Нажмите на кнопку **Удалить**. Система удалит все события из сравнения и отобразит соответствующее уведомление.

Ресурсы

В данном разделе описаны компоненты системы, которые обеспечивают настройку и управление различными функциями системы. Компоненты сгруппированы по их функциональным областям и включают коллекторы, модели данных и базы данных.

Содержание раздела:

- Работа с коллекторами
  - Работа с конвейерами
    - Работа с точками входа
    - Работа с VRL-трансформациями
    - Работа с фильтрами
    - Работа с маршрутизаторами
    - Работа с коррелляторами
    - Работа с нормализаторами
    - Работа с агрегаторами
    - Работа со сбором метрик
    - Работа с конечными точками
  - Работа с шинами
  - Работа с обогащениями
- Работа с моделями событий
  - Универсальная модель события
  - Типы данных
- Работа с базами данных событий
- Работа с хранилищами событий
- Работа с активными списками
- Работа с секретами
- Работа с доменами
  - Редактор связей
- Работа с переменными
- Работа с конфигурациями
- Работа с воркерами
- Работа со сканерами

Если в настройках сервера системы настроена интеграция с R-Vision Endpoint, в списке также отображается раздел **Агенты**.

Работа с коллекторами

Данный раздел описывает процесс работы с коллекторами в системе. Работа осуществляется в разделе **Ресурсы** → **Коллекторы** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О коллекторах

Коллектор — это компонент системы, который обеспечивает сбор событий из источников, их обработку и дальнейшую отправку на хранение и/или во внешние системы. События поступают в коллектор через установленные на нем конвейеры, посредством которых они могут быть обработаны.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новый коллектор.
- Кнопка ⤓ позволяет импортировать коллекторы в систему.
- Кнопка ⤒ позволяет экспортировать выбранные коллекторы из системы в формате JSON.
- Поле **Поиск** предназначено для быстрого поиска коллекторов в отображаемом списке по названию.
- Выпадающий список **Статус** позволяет настроить отображение списка коллекторов в соответствии с выбранным статусом.

В рабочей области отображается таблица имеющихся коллекторов. Таблица содержит следующие столбцы:

- **Название** — название коллектора, используемое для его идентификации в системе.
- **Статус** — текущий статус коллектора (включен/выключен).
- **Пространство** — название подключения к пространству, в котором расположен коллектор.
- **Конвейеров** — количество конвейеров в коллекторе.
- **Дата создания** — дата и время создания коллектора.
- **Дата изменения** — дата и время последнего изменения коллектора.
- **Описание** — описание коллектора.

При работе с таблицей коллекторов доступны следующие операции:

- Поиск коллекторов по полю **Название**.
- Фильтрация коллекторов по полю **Статус**. Статус выбирается из выпадающего списка.
- Сортировка коллекторов по полям **Название**, **Пространство**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

При выборе конкретного коллектора в правой части рабочей области отображается его карточка с детальной информацией. Карточка коллектора содержит следующие вкладки:

- **Информация** — общая информация о коллекторе и настройках логирования.
- **Конвейеры** — информация о конвейерах в коллекторе.
- **Шины** — информация о шинах в коллекторе.
- **Обогащение** — информация о таблицах обогащения, добавленных в коллектор.

Работа с коллектором

Доступные операции над коллекторами:

- Создание коллектора
- Просмотр коллектора
- Включение коллектора
- Выключение коллектора
- Изменение коллектора
- Удаление коллектора
- Импорт коллектора
- Экспорт коллектора

Создание коллектора

Чтобы создать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания коллектора.
3. Введите название коллектора.
4. Выберите подключение к пространству, в котором нужно создать коллектор. По умолчанию коллектор создается в центральном пространстве.
5. Укажите ресурсы, выделяемые для работы коллектора: количество ядер процессора в поле **CPU** и объем оперативной памяти в поле **RAM**. Значения можно вводить в полях или изменять с помощью кнопок + и —.

- При наведении курсора на поля **CPU** и **RAM** отображаются подсказки с количеством доступных ресурсов в каждом узле кластера Kubernetes.
- Ресурсы, выделенные на работу коллектора, не используются в распределенной корреляции.
- Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

6. При необходимости введите описание коллектора.
7. Для работы в режиме распределенной корреляции переведите переключатель **Распределенный коррелятор** в активное положение.

Изменение этой настройки потребует перезапуска коллектора.

Если система работает на базе кластера Kubernetes с одним узлом, включение режима распределенной корреляции недоступно.

8. Укажите количество ядер процессора (**CPU**) и объем оперативной памяти (**RAM**), выделяемый для работы распределенного коррелятора. Это можно сделать непосредственно в поле или с помощью кнопок + и —.

Операции полей `join` и `group` декларативного правила вычисляются на коррелляторе. Остальные поля, в том числе `alias` и `filter`, вычисляются на предпроцессорах. В составе распределенного коррелятора может быть только один коррелятор, но несколько предпроцессоров.

   a. Укажите ресурсы для коррелятора:
      i. Выберите узел для размещения коррелятора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.
      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы коррелятора. Количество доступных в узле ресурсов отображается под полями ввода.
   b. Укажите ресурсы для предпроцессора:
      i. Выберите узел для размещения предпроцессора. На выбор доступны только узлы кластера, относящиеся к пространству, в котором создается коллектор.
      ii. Укажите количество ядер процессора и объем оперативной памяти, выделяемые для работы предпроцессора. Количество доступных в узле ресурсов отображается под полями ввода.
9. Нажмите на кнопку **Добавить предпроцессор**, если необходимо добавить дополнительный предпроцессор.
10. При необходимости выберите уровень логирования из выпадающего списка **Уровень логирования**. Доступны следующие значения:
    - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.
    - **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
    - **Info** — записывается общая информация о работе и состоянии коллектора.
    - **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
    - **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.

11. Нажмите на кнопку **Создать**. Система создаст коллектор и отобразит соответствующее уведомление. Новый коллектор появится в списке раздела **Ресурсы** → **Коллекторы**.

Просмотр коллектора

Чтобы просмотреть коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.

В верхней части карточки отображается название коллектора. Справа от названия расположена кнопка ⋮, по которой открывается выпадающее меню с действиями, доступными для выбранного коллектора.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка коллектора содержит следующие вкладки:

- **Информация** — общая информация о коллекторе и настройках логирования.
- **Конвейеры** — информация о конвейерах в коллекторе.
- **Шины** — информация о шинах в коллекторе.
- **Обогащение** — информация о таблицах обогащения, добавленных в коллектор.

Вкладка Информация

На вкладке **Информация** отображаются основные сведения о коллекторе, сгруппированные по следующим секциям:

- **Метрики коллектора**:
  - **CPU** — количество ядер и уровень загруженности процессора сервиса.
  - **RAM** — объем и уровень загруженности оперативной памяти сервиса.
  - **HDD** — объем и уровень загруженности жесткого диска сервиса.
- **Метрики распределенного коррелятора**:
  - Характеристика процессора (CPU), объем оперативной памяти (RAM) и жесткого диска (HDD) для каждого узла распределенного коррелятора, включая данные по каждому предпроцессору и коррелятору.
- **Информация**:
  - **ID** — уникальный идентификационный код коллектора, генерируемый автоматически при его создании.
  - **Название** — имя коллектора, используемое для его идентификации в системе.
  - **Пространство** — название подключения к пространству, в котором расположен коллектор.

Чтобы получить подробные сведения о подключении, нажмите на его

название. Система откроет раздел **Настройки → Пространства и сервисы → Пространства** и отобразит на экране подробную карточку этого подключения.

- **ID тенанта** — уникальный идентификационный код тенанта, к которому относится коллектор.
- **Статус** — текущий статус коллектора (включен, выключен или ошибка).
- **IP-адрес** — IP-адрес, с которым связан коллектор.
- **Имя хоста** — имя хоста, с которым связан коллектор.
- **Описание** — краткое описание функциональности коллектора.
- **Конвейеров** — количество конвейеров, связанных с коллектором.
- **Дата создания** — дата и время создания коллектора.
- **Создал** — пользователь, инициировавший создание коллектора.
- **Дата изменения** — дата и время последнего изменения коллектора.
- **Изменил** — пользователь, выполнивший последнее изменение коллектора.

- **Просмотр логов**:
  - **Уровень логирования** — уровень логирования коллектора: **Error**, **Warn**, **Info**, **Debug**, **Trace** или **По умолчанию**.

  В секции также расположена кнопка **Просмотреть логи**, позволяющая открыть окно для просмотра логов коллектора.

- **EPS за последние 30 минут** — график с количеством зарегистрированных событий в секунду (EPS) за последние 30 минут с шагом в 5 минут для текущего коллектора и всех коллекторов системы.
- **Трафик за последние 30 минут** — график с объемом входящего трафика, полученного текущим коллектором и всеми коллекторами системы, за последние 30 минут с шагом в 5 минут.

В верхней части карточки также расположен переключатель статуса коллектора.

Вкладка Конвейеры

На вкладке **Конвейеры** отображается список конвейеров, добавленных в коллектор.

По умолчанию для каждого конвейера отображается только его название и цветовой индикатор статуса конфигурации:

- зеленый — конфигурация установлена;
- серый — конфигурация не установлена;
- красный — ошибка установки.

Также в строке конвейера имеется кнопка действий (⋮). При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранного конвейера.

Чтобы получить более подробные сведения о конвейере, нажмите на кнопку ⌄ в правой части его строки. Для конвейера отобразится карточка со следующей информацией:

- **ID** — уникальный идентификационный код конвейера, генерируемый автоматически при его создании.
- **Описание** — краткое описание функциональности конвейера.
- **Статус конфигурации** — текущий статус конфигурации конвейера: **Установлена**, **Не установлена** или **Ошибка установки**.
- **Дата создания** — дата и время создания конвейера.
- **Создал** — пользователь, инициировавший создание конвейера.
- **Дата изменения** — дата и время последнего изменения конвейера.
- **Изменил** — пользователь, выполнивший последнее изменение конвейера.
- **EPS за последние 30 минут** — график с количеством зарегистрированных событий в секунду (EPS) за последние 30 минут с шагом в 5 минут.
- **Адреса точек входа** — адреса всех точек входа, добавленных на конвейер.

  Для каждой точки входа в правой части строки расположена кнопка . При нажатии на эту кнопку система скопирует адрес точки входа в буфер обмена и отобразит соответствующее уведомление.

В карточке конвейера также расположены кнопки для настройки и установки или отключения его конфигурации.

В верхней части вкладки содержится поле для быстрого поиска конвейеров по названию.

В нижней части вкладки расположены кнопки для добавления и импорта конвейеров.

Вкладка Шины

На вкладке **Шины** отображается список шин в коллекторе.

По умолчанию для каждой шины отображается только ее название. Также в строке шины имеется кнопка действий (⋮). При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранной шины.

Чтобы получить более подробные сведения о шине, нажмите на кнопку ⌄ в правой части ее строки. Для шины отобразится карточка со следующей информацией:

- **ID** — уникальный идентификационный код шины, генерируемый автоматически при ее создании.
- **Описание** — краткое описание функциональности шины.
- **Дата создания** — дата и время создания шины.
- **Создал** — пользователь, инициировавший создание шины.
- **Дата изменения** — дата и время последнего изменения шины.
- **Изменил** — пользователь, выполнивший последнее изменение шины.

В верхней части вкладки содержится поле для быстрого поиска шин по названию.

В нижней части вкладки расположена кнопка для добавления шин.

Вкладка Обогащение

На вкладке **Обогащение** отображается список обогащений в коллекторе.

По умолчанию для каждого обогащения отображается только его название. Также в строке обогащения имеется кнопка действий (⋮). При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранного обогащения.

Чтобы получить более подробные сведения об обогащении, нажмите на кнопку ⌄ в правой части его строки. Для обогащения отобразится карточка со следующей информацией:

- **ID обогащения** — уникальный идентификационный код компонента обогащения, генерируемый автоматически при добавлении его в коллектор.
- **Описание** — краткое описание функциональности обогащения.
- **ID таблицы обогащения** — уникальный идентификационный код таблицы обогащения, которая была использована при создании обогащения.
- **Таблица обогащения** — название таблицы обогащения, которая была использована при создании обогащения.
- **Версия таблицы обогащения** — версия таблицы обогащения, указанная в поле `version` в настройках таблицы.
- **Дата создания** — дата и время создания обогащения.
- **Создал** — пользователь, инициировавший создание обогащения.
- **Дата изменения** — дата и время последнего изменения обогащения.
- **Изменил** — пользователь, выполнивший последнее изменение обогащения.

В верхней части вкладки содержится поле для быстрого поиска обогащений по названию.

В нижней части вкладки расположена кнопка для добавления обогащений.

Включение коллектора

Для того чтобы коллектор мог начать работу с событиями, поступающими в систему, его нужно включить.

Вместе с коллектором также выключаются и все связанные с ним конвейеры.

Чтобы включить коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Убедитесь, что в карточке коллектора открыта вкладка **Информация**.
4. Переведите переключатель состояния коллектора в верхней части карточки в активное положение. Отобразится окно подтверждения включения коллектора.
5. Нажмите на кнопку **Включить**. Система включит коллектор и отобразит соответствующее

уведомление.

Если при включении коллектора возникли ошибки, система отобразит сообщение об этом. Для просмотра списка ошибок нажмите на кнопку **Показать**.

Выключение коллектора

Чтобы выключить работающий коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Убедитесь, что в карточке коллектора открыта вкладка **Информация**.
4. Переведите переключатель в верхней части коллектора в неактивное положение. Отобразится окно подтверждения выключения коллектора.
5. Нажмите на кнопку **Выключить**. Система выключит коллектор и отобразит соответствующее уведомление.

Изменение коллектора

Чтобы изменить коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек коллектора.
4. Внесите изменения в требуемые поля коллектора.
5. Нажмите на кнопку **Сохранить**. Система сохранит измененные данные коллектора и отобразит соответствующее уведомление.

Изменить название и описание коллектора можно непосредственно из его карточки:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Убедитесь, что в карточке коллектора открыта вкладка **Информация**.
4. Нажмите на кнопку рядом с текущим названием или описанием коллектора в его карточке. Система отобразит поле ввода значения.
5. Введите новое значение поля.
6. Нажмите **Сохранить**. Система сохранит изменения и отобразит

соответствующее уведомление. Обновленная информация отобразится в карточке и списке коллекторов.

Удаление коллектора

Удаление недоступно для коллекторов, используемых в других сущностях системы.

Вместе с коллектором также удаляются и все добавленные в него конвейеры, шины и обогащения.

Чтобы удалить коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления коллектора.
4. Нажмите на кнопку **Удалить**. Система удалит коллектор и отобразит соответствующее уведомление. Удаленный коллектор будет исключен из списка.

Импорт коллектора

Для загрузки конфигурации коллектора в систему применяется функционал импорта, поддерживающий обработку файлов в формате JSON.

Перед тем как импортировать коллектор:

1. Создайте или импортируйте в систему хранилище событий.
2. Импортируйте в систему все элементы экспертизы, которые содержит импортируемый коллектор.

Чтобы импортировать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на кнопку ⤓ на панели инструментов. Система отобразит окно импорта коллектора.
3. Нажмите **Выбрать файл**. Откроется окно проводника операционной системы.
4. В окне выберите файл для импорта и нажмите на кнопку **Открыть**. Файл отобразится в окне импорта коллектора.
5. Нажмите на кнопку **Импортировать**. Коллектор будет загружен в систему. Система отобразит окно с итогами импорта.

Экспорт коллектора

Чтобы экспортировать коллектор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Экспортировать**. Отобразится окно экспорта данных.
4. Нажмите на кнопку **Экспортировать**. Система загрузит данные коллектора на устройство пользователя в формате JSON и отобразит соответствующее уведомление.

Вы также можете экспортировать сразу несколько коллекторов. Для этого:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Установите флажки напротив тех коллекторов, которые вы хотите экспортировать, в левом столбце таблицы.
3. Нажмите на кнопку ⬆ на панели инструментов. Система запросит подтверждение выполнения экспорта.
4. Нажмите на кнопку **Экспортировать**. Система загрузит данные коллекторов на устройство пользователя в формате JSON и отобразит соответствующее уведомление.

Также можно экспортировать конфигурацию коллектора из объектного хранилища NATS ObjectStore.

Работа с логами коллектора

Логирование коллекторов обеспечивает запись событий и состояний системы сбора данных, что позволяет администраторам и разработчикам отслеживать и анализировать работу системы. Уровень логирования настраивается вручную при создании коллектора или изменении его параметров, что позволяет гибко управлять объемом и детализацией регистрируемых данных. Доступные уровни логирования включают в себя Error, Warn, Info, Debug и Trace, каждый из которых предоставляет различный уровень информации о работе коллектора.

Функционал просмотра логов доступен через пользовательский интерфейс на вкладке **Информация** карточки коллектора. Здесь отображается текущий уровень логирования и предоставляется возможность просмотра самих логов.

Доступные операции над логами коллектора:

- Просмотр логов коллектора
- Фильтрация списка логов коллектора
- Копирование логов коллектора
- Изменение уровня логирования

Логи коллектора очищаются при его выключении.

Просмотр логов коллектора

Чтобы просмотреть логи коллектора:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Убедитесь, что в карточке коллектора открыта вкладка **Информация**.
4. В секции **Просмотр логов** система отображает текущий уровень логирования коллектора. Нажмите на кнопку **Просмотреть логи**. Система отобразит окно со списком логов коллектора.

В списке логов доступен просмотр только последних 100 000 записей.

В левом верхнем углу окна отображается ID просматриваемого коллектора.

Доступны два режима вывода логов:

- **Режим мониторинга** — вывод логов в реальном времени.
- **Режим поиска по временному периоду** — вывод логов в рамках заданного периода. Максимальный период запроса — 30 дней.

Переключатель **Мониторинг** позволяет управлять режимом вывода логов.

- Логи коллектора можно также просмотреть:
  - при нажатии на кнопку **Просмотреть логи** () на странице конфигурации конвейера;
  - при нажатии на кнопку **Посмотреть логи** в карточке сервиса;
  - в кластере Kubernetes.
- Если ошибки в коллекторе не удается устранить с помощью анализа его логов, рекомендуется проанализировать логи пода `collector-manager`, управляющего работой всех коллекторов в системе.

Фильтрация списка логов коллектора

Просмотр логов за конкретный временной интервал недоступен в режиме мониторинга.

Чтобы отфильтровать список логов коллектора в режиме просмотра логов:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Убедитесь, что в карточке коллектора открыта вкладка **Информация**.

4. В секции **Просмотр логов** система отображает текущий уровень логирования коллектора. Нажмите на кнопку **Просмотреть логи**. Система отобразит окно со списком логов коллектора.
5. Переведите переключатель **Мониторинг** в неактивное положение. Настройки фильтрации станут доступными для редактирования.
6. Настройте период поиска логов. Для этого нажмите на кнопку и выберите из выпадающего списка способ задания периода:
   - **Последний период** — задайте значение вручную или с помощью кнопок — и +, после чего выберите единицы времени из выпадающего списка. Поиск логов будет выполняться в указанном диапазоне вплоть до нынешнего момента.
   - **Задать период** — укажите даты начала и конца периода, за который будет выполняться поиск логов.
   - **Часто используемые** — выберите в разделе один из предлагаемых периодов.
7. Нажмите на кнопку **Найти**. Список логов, соответствующий заданному периоду, отобразится в окне.

Копирование логов коллектора

Чтобы скопировать записи журнала логов коллектора:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Убедитесь, что в карточке коллектора открыта вкладка **Информация**.
4. В секции **Просмотр логов** система отображает текущий уровень логирования коллектора. Нажмите на кнопку **Просмотреть логи**. Система отобразит окно со списком логов коллектора.
5. Нажмите на кнопку **Скопировать** в правой нижней части экрана. Система скопирует записи журнала в буфер обмена и отобразит соответствующее уведомление.

Изменение уровня логирования

Изменение уровня логирования выполняется через редактирование настроек коллектора.

Чтобы изменить уровень логирования коллектора:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек коллектора.
4. Выберите значение из выпадающего списка **Уровень логирования**. Доступны следующие значения:
   - **Error** — записываются критические ошибки, которые могут влиять на работоспособность коллектора или препятствовать работе коллектора.

- **Warn** — записываются предупреждения о потенциальных проблемах, которые могут требовать внимания, но не являются критическими.
- **Info** — записывается общая информация о работе и состоянии коллектора.
- **Debug** — записываются подробные отладочные сообщения, которые могут использоваться для диагностики и устранения проблем с коллектором.
- **Trace** — самый подробный уровень логирования, предоставляющий внутреннюю информацию о каждом шаге работы коллектора.

5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке коллектора.

Работа с конвейерами

Данный раздел описывает процесс работы с конвейерами обработки событий в системе. Работа осуществляется на вкладке **Конвейеры** в карточке коллектора.

О конвейерах

Конвейер — компонент коллектора, представляющий собой упорядоченную последовательность взаимосвязанных элементов, выполняющих различные этапы обработки событий. В процессе работы конвейера происходит прием событий, их нормализация, перенаправление, обогащение данными, фильтрация, агрегация, корреляционный анализ и отправка как отдельных, так и сгруппированных или коррелированных событий для хранения и/или передачи во внешние системы.

Список конвейеров коллектора отображается на вкладке **Конвейеры** в его карточке. Над списком расположено поле для поиска конвейеров по названию.

Доступные операции над конвейером:

- Добавление конвейера
- Просмотр конвейера
- Настройка конфигурации конвейера
- Установка конфигурации конвейера
- Изменение конвейера
- Создание копии конвейера
- Удаление конвейера
- Импорт конвейера
- Экспорт конвейера

Добавление конвейера

Чтобы добавить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку

этого коллектора с подробной информацией о нем.

3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. В нижней части окна нажмите на кнопку **Добавить**. На экране отобразится окно добавления конвейера.
5. Введите название конвейера.
6. При необходимости введите описание конвейера.

Установите флажок **Открыть конфигурацию конвейера**, чтобы сразу перейти в окно настроек конфигурации конвейера после его создания.

7. Нажмите на кнопку **Добавить**. Система создаст конвейер и отобразит соответствующее уведомление. Новый конвейер появится в списке на вкладке **Конвейеры** в карточке коллектора.

Просмотр конвейера

Чтобы отобразить информацию о конвейере:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.

В верхней части карточки отображается название конвейера. Слева от названия расположен цветовой индикатор статуса конфигурации конвейера:

- зеленый — конфигурация установлена;
- серый — конфигурация не установлена.

Справа от названия конвейера расположена кнопка ⋮ , по которой открывается выпадающее меню с действиями, доступными для выбранного конвейера.

Карточка конвейера содержит следующую информацию:

- **ID** — уникальный идентификационный код конвейера, генерируемый автоматически при его создании.
- **Описание** — краткое описание функциональности конвейера.
- **Статус конфигурации** — текущий статус конфигурации конвейера: **Установлена** или **Не установлена**.
- **Дата создания** — дата и время создания конвейера.
- **Создал** — пользователь, инициировавший создание конвейера.
- **Дата изменения** — дата и время последнего изменения конвейера.

- **Изменил** — пользователь, выполнивший последнее изменение конвейера.
- **EPS за последние 30 минут** — график с количеством зарегистрированных событий в секунду (EPS) за последние 30 минут с шагом в 5 минут.
- **Адреса точек входа** — адреса всех точек входа, добавленных на конвейер.

  Для каждой точки входа в правой части строки расположена кнопка . При нажатии на эту кнопку система скопирует адрес точки входа в буфер обмена и отобразит соответствующее уведомление.

В карточке конвейера также расположены кнопки для настройки и установки или отключения его конфигурации.

Настройка конфигурации конвейера

При настройке конвейера необходимо создать его компоненты, сконфигурировать их, упорядочить и объединить в последовательность, соответствующую логике обработки данных.

Для базовой обработки и хранения данных в системе достаточно настроить точку входа и конечную точку. Дополнительные элементы конвейера, такие как фильтр, нормализатор и коррелятор, позволяют настроить более сложную логику обработки данных, проходящих через конвейер событий.

Эти элементы можно разместить между точкой входа и конечной точкой в любой необходимой последовательности и комбинации. При этом можно использовать несколько элементов одного типа для более детализированной настройки обработки.

Для управления порядком и логикой прохождения данных через различные этапы конвейера можно применять маршрутизатор. Он позволяет перенаправлять и распределять поток данных в соответствии с заранее заданными условиями.

Чтобы настроить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку ( ) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации, над которой расположены кнопки установки/отключения конфигурации конвейера.

- Диаграмму конфигурации также можно открыть, нажав в строке конвейера в карточке коллектора кнопку действий ( ) и выбрав опцию **Конфигурация конвейера**.
- По умолчанию справа от переключателя состояния конвейера система показывает уведомление о трех выявленных ошибках из-за отсутствия

элементов на конвейере. Для просмотра всех ошибок нажмите **Показать**.

6. Настройте конфигурацию конвейера:
   a. Создайте и настройте элементы конвейера.
   b. Свяжите элементы конвейера в определенной последовательности согласно желаемой логике получения, обработки и передачи данных.
7. Установите конфигурацию конвейера.

Если при настройке конвейера были допущены ошибки, система отобразит сообщение об этом справа от кнопки установки/отключения конфигурации конвейера. Для просмотра списка ошибок нажмите на кнопку **Показать**.

Интерфейс конфигурации конвейера

Редактирование конфигурации происходит на визуальной диаграмме через добавление элементов и связей между ними. Каждый элемент конвейера отвечает за конкретный этап обработки событий.

На диаграмме отображаются типы и названия элементов, а также дополнительная информация, которая зависит от типа элемента:

- **Точка входа**:
  - тип точки;
  - адрес подключения (при наличии);
  - порт подключения (при наличии);
  - тип протокола (при наличии);
  - уровни угрозы (для точки входа типа Audit).
- **Коррелятор**: количество правил корреляции.
- **Нормализатор**: количество правил нормализации.
- **Агрегатор**: количество правил агрегации.
- **Шина**: тип соединения с шиной (**Получение** или **Отправка**).
- **Конечная точка**: тип точки, а также, в зависимости от типа:
  - для конечной точки **Хранилище событий**: название хранилища;
  - для конечной точки **Внешняя система**: тип внешней системы и ее адрес (при наличии).

Если информация сокращена, при наведении курсора на элемент отображается всплывающее сообщение с полным текстом.

Каждый элемент имеет хотя бы один вход или выход, при этом вход может быть только один, а выходов может быть несколько. На диаграмме конфигурации входы и выходы элемента обозначены метками на его боковых сторонах. Вход соответствует метке на левой стороне элемента, а выходы — метке на правой стороне.

Элементы конвейера соединены между собой направляющими линиями, которые указывают порядок и направление обработки событий. Элементы можно располагать между точкой входа и

конечной точкой или шиной в любой желаемой последовательности и комбинации. Например, можно использовать несколько элементов одного типа.

Доступны два режима работы с конфигурацией конвейера:

- **Актуальная версия** — доступен только просмотр конфигурации конвейера.
- **Черновик** — доступно редактирование конфигурации конвейера.

Редактирование конфигурации в режиме черновика доступно даже при установленной конфигурации.

Чтобы установить режим работы с конфигурацией конвейера, выберите его из выпадающего списка **Версия**, который расположен в правой части панели инструментов.

В режиме черновика на панели инструментов доступны следующие кнопки:

- **Добавить элемент** — позволяет добавить новый элемент конвейера.
- **Сбросить изменения** ( ) — позволяет сбросить изменения в черновике до сохраненной ранее конфигурации. Если изменений в черновике не было, кнопка не активна.
- **Отцентровать конвейер** ( ) — позволяет автоматически отмасштабировать и отцентровать диаграмму без упорядочивания ее элементов.
- **Упорядочить элементы** ( ) — позволяет автоматически упорядочить элементы, а также отмасштабировать и отцентровать диаграмму по ее элементам.
- **Просмотреть логи** ( ) — позволяет просмотреть логи коллектора.
- **Установить конфигурацию** — позволяет установить настроенную в черновике конфигурацию конвейера.
- **Отключить конфигурацию** — позволяет отключить текущую установленную конфигурацию конвейера.

В режиме актуальной версии на панели инструментов доступны следующие кнопки:

- **Посмотреть логи** ( ) — позволяет просмотреть логи коллектора.
- **Сбросить метрики** ( ) — позволяет сбросить текущие метрики у элементов конвейера.
- **Отключить конфигурацию** — позволяет отключить текущую установленную конфигурацию конвейера.

- Чтобы приблизить или отдалить элементы на диаграмме, используйте колесико мыши.
- Чтобы перемещаться по диаграмме, зажмите кнопку мыши и перемещайте курсор в нужном направлении.

Также в режиме актуальной версии в нижней части каждого элемента конвейера отображаются следующие метрики:

- **Получено событий** ( ) — количество событий, полученных элементом за последние 30 минут;
- **Отправлено событий** ( ) — количество событий, отправленных элементом за последние 30

минут;

- **Ошибки** (⚠) — количество ошибок на элементе за последние 30 минут.

Дополнительные метрики можно посмотреть во вкладке **Метрики** окна просмотра элемента конвейера.

Добавление элемента конвейера

Каждый элемент конвейера соответствует определенному этапу конвейера обработки событий. Добавление новых элементов конвейера доступно только в режиме черновика.

Доступные типы элементов:

- **Точка входа** — имеет один выход и не имеет входов. Подключается к источнику данных и на выходе выдает поток исходных событий, поступающих из источника. Выбор типа точки входа зависит от формата получаемых данных.
- **VRL-трансформация** — имеет один вход и один выход. На входе принимает события и проводит их преобразование. На выходе выдает преобразованные события. Инструкция для трансформации событий указывается с использованием языка VRL.

  Рекомендуется для несложных модификаций в структуре событий: например, для использования одной или нескольких функций, упрощающих дальнейшую работу с событиями, или добавления/изменения/удаления поля в структуре.

- **Маршрутизатор** — имеет один вход и несколько выходов: по одному выходу на каждый фильтр (маршрут), задаваемый на языке VRL. На входе проверяет каждое событие по условиям маршрутов и направляет его на выходы тех маршрутов, где условие фильтрации выполняется.
- **Фильтр** — имеет один вход и один выход. Содержит условие фильтрации, задаваемое на языке VRL. На входе принимает событие, проверяет его по условию фильтрации и направляет на выход только в том случае, если условие выполняется. В противном случае событие отсеивается и не участвует в дальнейшей обработке.
- **Шина** — работает в рамках одного коллектора. В зависимости от типа соединения на конфигурации конвейера, имеет либо только один вход без выходов (получение), либо только один выход без входов (отправка). Принимает события на входе соединения типа **Получение** и пересылает их на выход соединения типа **Отправка**.

  Шины позволяют настроить обмен событиями между несколькими конвейерами.

- **Коррелятор** — имеет один вход и один выход. На входе принимает поток событий и применяет к ним правила корреляции, определяя взаимосвязанные события, которые соответствуют заданным критериям. На выходе выдает корреляционные события.
- **Нормализатор** — имеет один вход и один выход. На входе принимает поток событий и применяет к ним правила нормализации, осуществляя разбор и преобразование полученных полей события в нужную структуру. На выходе выдает нормализованные события.

  Рекомендуется для проведения сложных и объемных преобразований, применения одних и тех же преобразований в нескольких конвейерах и возможности изменения правил преобразований без перезапуска конвейера.

- **Конечная точка** — имеет один вход и не имеет выходов. На входе принимает событие и, в зависимости от заданных настроек, пересылает его в хранилище событий или во внешнюю систему. Выбор типа конечной точки зависит от пункта назначения, в который пересылаются события.
- **Агрегатор** — имеет один вход и один выход. На входе принимает поток событий и применяет к ним правила агрегации, объединяя события в группы. На выходе выдает агрегированные события, которые представляют собой совокупности характеристик исходных событий, удовлетворяющих условиям агрегации.
- **Сбор метрик** — имеет один выход и не имеет входов. Собирает метрики по выбранным элементам конвейеров в рамках коллектора и на выходе выдает поток событий, содержащий собранные метрики.

Чтобы добавить элемент конвейера:

1. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Черновик**.
2. Нажмите **Добавить элемент** над полем диаграммы.
3. Введите необходимые параметры и нажмите на кнопку **Добавить**. Новый элемент отобразится на диаграмме конфигурации.

Добавленные элементы не учитываются по умолчанию в текущей установленной конфигурации. Чтобы подгрузить изменения в установленную конфигурацию, ее требуется установить заново.

Настройка связей между элементами конвейера

Настройка связей между элементами конвейера доступна только в режиме черновика.

Чтобы связать элементы, наведите курсор на метку, расположенную на боковой стороне элемента. Курсор примет вид крестика. Перетащите крестик к метке на другом элементе. Связь будет создана, порядок обработки событий будет определен логикой связи.

Один вход может быть соединен с несколькими выходами, а один выход — с несколькими входами. При этом недопустимы циклические связи.

Чтобы удалить связь между элементами, выделите ее и нажмите клавишу DELETE. В Windows также можно использовать клавишу BACKSPACE.

Настроенные связи не учитываются по умолчанию в текущей установленной конфигурации. Чтобы подгрузить изменения в установленную конфигурацию, ее требуется установить заново.

Просмотр элемента конвейера

Просмотр элементов конвейера доступен только в режиме актуальной версии конфигурации.

Чтобы просмотреть элемент конвейера:

1. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Актуальная**.
2. Нажмите на кнопку **Просмотр** ( ) справа от названия элемента. Отобразится окно просмотра элемента.

В окне доступны две вкладки:

- **Информация** — отображает заданные настройки элемента конвейера.
- **Метрики** — отображает сведения о полученных и отправленных событиях, об ошибках, а также о загруженности элемента в виде соответствующих графиков. Отображаются сведения за последние 30 минут. Шаг графика — 15 секунд.

  Доступны следующие графики:

  - **Получено событий**;
  - **Отправлено событий**;
  - **Получено байт**;
  - **Отправлено байт**;
  - **Количество ошибок**;
  - **Утилизация компонента**.

Метрика утилизации характеризует степень загруженности элемента. Она принимает значения от 0 до 1:

- чем ближе утилизация к 1, тем меньше время простоя элемента;
- если 0, то элемент не используется;
- если 1, то элемент никогда не простаивает.

Для элемента типа **Конечная точка** дополнительно доступны графики **Буферизировано событий** и **Буферизировано байт**, отображающие статистику буферизации.

Для нормализаторов и коррелляторов на вкладке **Информация** доступна кнопка **Изменить** для изменения заданных настроек.

Изменение элемента конвейера

Изменение элементов конвейера доступно только в режиме черновика.

Чтобы изменить элемент конвейера:

1. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Черновик**.
2. Откройте окно настроек элемента одним из следующих способов:
   - Нажмите на кнопку действий (⋮) справа от названия элемента и выберите опцию **Изменить**.
   - Дважды нажмите на элемент на диаграмме конфигурации конвейера.
3. Измените требуемые настройки элемента.
4. Нажмите на кнопку **Сохранить**. Система обновит информацию об элементе.

Изменение нормализаторов и коррелляторов доступно также и в режиме актуальной версии конфигурации конвейера:

1. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Актуальная**.
2. Нажмите на кнопку **Просмотр** ( ) справа от названия элемента. Отобразится окно просмотра элемента.
3. Удостоверьтесь, что в окне просмотра элемента открыта вкладка **Информация**.
4. Нажмите на кнопку **Изменить**. Отобразится окно настройки элемента.
5. Измените требуемые настройки элемента.
6. Нажмите на кнопку **Сохранить**. Система обновит информацию об элементе.

Измененные настройки элементов не учитываются по умолчанию в текущей установленной конфигурации. Чтобы подгрузить изменения в установленную конфигурацию, ее требуется установить заново.

Удаление элемента конвейера

Удаление элементов конвейера доступно только в режиме черновика.

Чтобы удалить элемент конвейера:

1. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Черновик**.
2. Нажмите на кнопку действий (⋮) справа от названия элемента и выберите опцию **Удалить**. Отобразится окно подтверждения удаления.
3. Нажмите на кнопку **Удалить**. Элемент конвейера будет удален.

Удаленные элементы не учитываются по умолчанию в текущей установленной конфигурации. Чтобы подгрузить изменения в установленную конфигурацию, ее требуется установить заново.

Валидация конвейера

При настройке конвейера система проверяет конфигурацию на наличие ошибок. Сообщение о количестве допущенных ошибок отображается справа от кнопки установки/отключения конфигурации конвейера. Чтобы просмотреть список ошибок, нажмите на кнопку **Показать**.

Могут возникать два типа ошибок: ошибки конфигурации конвейера в целом и ошибки отдельных элементов. Ошибки конвейера включают в себя:

- Отсутствие хотя бы одного источника событий: точки входа, шины на получение событий или сбора метрик.

Для отладки конвейера вы можете использовать точку входа Demo Logs, которая генерирует случайные события.

- Отсутствие хотя бы одной конечной точки или шины на отправку событий.

Для отладки конвейера вы можете использовать конечную точку Blackhole, которая не сохраняет и не отправляет события.

Ошибки элементов связаны с некорректной настройкой конкретных элементов. Чтобы перейти к проблемному элементу, под описанием ошибки нажмите на кнопку **К элементу**. Ошибки элементов включают:

- Отсутствие связей на входе элемента.
- Отсутствие связей на выходе элемента.
- Невалидный код VRL.
- Неуникальное значение порта в точке входа.
- Правило, содержащее отсутствующий активный список.
- Правило, содержащее отсутствующую таблицу обогащения.
- Циклическая связь элементов. Ошибка возникает, если элемент связан сам с собой через другие элементы.
- Отсутствие необходимых ресурсов. Ошибка возникает, если в конфигурации элементе импортированного конвейера указаны ресурсы, которые отсутствуют в системе. Это могут быть:
  - Хранилище событий в конечной точке;
  - Секреты в точке входа или конечной точке.

Сброс метрик

Чтобы сбросить метрики элементов конвейера:

1. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Текущая**.
2. Нажмите на кнопку **Сбросить метрики** ( ) на панели инструментов. Отобразится окно подтверждения сброса метрик.
3. Нажмите на кнопку **Сбросить**. Система сбросит текущие метрики у элементов конвейера и графики метрик по компонентам до 0 и отобразит соответствующее уведомление.

При сбросе метрик данные на графиках **EPS за последние 30 минут** в карточке коллектора и карточке конвейера не обнуляются. На графиках отражается общий показатель.

Установка конфигурации конвейера

Установка конфигурации конвейера доступна только в режиме черновика.

Для того чтобы установить конфигурацию конвейера в коллекторе, убедитесь, что связанный с ним коллектор уже включен.

Чтобы установить конфигурацию конвейера в коллекторе:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.

3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации, над которой расположены кнопки установки/отключения конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов выбран вариант **Черновик**.
7. Нажмите на кнопку **Установить конфигурацию** над диаграммой конфигурации. Отобразится окно подтверждения установки конфигурации.
8. Нажмите на кнопку **Установить**. Система установит конфигурацию конвейера и отобразит соответствующее уведомление. События начнут поступать в систему и обрабатываться согласно настроенной в черновике конфигурации конвейера.

Отключение активной конфигурации доступно из любого режима работы с конвейером.

Чтобы отключить активную конфигурацию конвейера, нажмите на кнопку **Отключить конфигурацию** над диаграммой конфигурации и подтвердите отключение. Система отключит конвейер и отобразит соответствующее уведомление. События перестанут поступать в систему и нормализовываться.

Чтобы выйти из диаграммы конфигурации и вернуться на вкладку **Конвейеры** в карточке коллектора, нажмите на кнопку **Назад** в левой верхней части экрана.

Если диаграмма конфигурации была открыта через вкладку **Конвейеры** в карточке элемента экспертизы, то по нажатии на кнопку **Назад** осуществится переход обратно в эту вкладку.

Включить и выключить конвейер также можно в карточке конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Включите/выключите конвейер:
   - Для включения конвейера используйте кнопку **Установить конфигурацию** в нижней части карточки.
   - Для выключения конвейера используйте кнопку **Отключить конфигурацию** в нижней части карточки.

Изменение конвейера

Перед тем как редактировать конфигурацию конвейера, убедитесь, что его

конфигурация не установлена в коллекторе.

Чтобы изменить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку действий (⋮) в строке конвейера и выберите опцию **Изменить**. Система отобразит окно редактирования конвейера.
5. Внесите изменения в требуемые поля.
6. Нажмите на кнопку **Сохранить**. Измененные данные конвейера будут сохранены.

Создание копии конвейера

Чтобы создать копию конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку действий (⋮) в строке конвейера и выберите опцию **Создать копию**. Система отобразит окно создания копии конвейера.
5. Измените требуемые поля конвейера.
6. Нажмите на кнопку **Создать**. Копия конвейера появится в списке конвейеров на вкладке **Конвейеры** в карточке коллектора.

Удаление конвейера

Удаление недоступно для конвейеров, используемых в других сущностях системы.

Чтобы удалить конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку действий (⋮) в строке конвейера и выберите опцию **Удалить**. Система отобразит окно подтверждения действия.

5. Нажмите на кнопку **Удалить**. Система удалит выбранный конвейер и отобразит соответствующее уведомление. Удаленный конвейер исчезнет из списка конвейеров на вкладке **Конвейеры** в карточке коллектора.

Вместе с конвейером также удаляются и все добавленные в него элементы.

Импорт конвейера

Чтобы импортировать конвейер:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку **Импортировать** в нижней части карточки. Отобразится окно импорта конвейера.
5. Нажмите на кнопку **Выбрать файл** и выберите файл конвейера для импорта.
6. Нажмите на кнопку **Загрузить**. Загруженный конвейер отобразится в списке конвейеров на вкладке **Конвейеры** в карточке коллектора.

Экспорт конвейера

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на кнопку действий (⋮) в строке конвейера и выберите опцию **Экспортировать**. Система отобразит окно подтверждения действия.
5. Нажмите на кнопку **Экспортировать**. Данные конвейера будут экспортированы в файл формата JSON.

Работа с точками входа

Данный раздел описывает процесс работы с точками входа на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О точках входа

Точка входа — это элемент конвейера, который отвечает за доставку в систему исходных событий. Выбор типа точки входа зависит от формата данных, получаемых из источника события. На диаграмме конфигурации конвейера точка входа имеет один выход и не имеет входов.

В системе доступны следующие типы точек входа:

- AMQP
- Audit
- Database
- Demo Logs
- HTTP Client
- HTTP Server
- Kafka
- Logstash
- Netflow
- R-Vision Endpoint
- Socket
- Syslog
- Vector
- WMI
- NATS JetStream
- Global Bus
- R-Vision EVO Endpoint

Доступные операции над точкой входа:

- Добавление точки входа
- Изменение точки входа
- Удаление точки входа
- Настройка точки входа

Добавление точки входа

Чтобы добавить точку входа на конфигурацию конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.

7. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Точка входа**. Отобразится окно добавления точки входа.
8. При необходимости выберите шаблон для автоматического заполнения полей точки входа. По умолчанию выбран вариант **Без шаблона**.
9. Заполните поля (набор и содержание полей могут отличаться в зависимости от типа точки):
    - Название точки входа.
    - Тип точки входа.

      Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. Если планируется размещение точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

    - Политика аудита источников.

      Точка входа может быть привязана только к одной политике аудита источников, но к одной политике может быть привязано несколько точек входа.

    - Формат точки входа.
    - Данные для подключения.
    - Настройки шифрования соединения (опционально).
    - Количество событий (опционально).
    - Интервал в секундах (опционально).
10. Установите флажок **Сохранить как шаблон**, если настройку точки входа необходимо сохранить в качестве шаблона для дальнейшего использования.
11. Нажмите **Добавить**. Новая точка входа отобразится на диаграмме конфигурации.

Изменение точки входа

Чтобы изменить точку входа:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.

7. Откройте окно настроек точки входа одним из следующих способов:
   - Нажмите на кнопку действий (⋮) справа от названия точки входа и выберите опцию **Изменить**.
   - Дважды нажмите на точку входа на диаграмме конфигурации конвейера.
8. Внесите изменения в требуемые поля.
9. Нажмите на кнопку **Сохранить**. Измененные данные точки входа будут сохранены.

Удаление точки входа

Чтобы удалить точку входа:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку действий (⋮) справа от названия точки входа и выберите опцию **Удалить**. Система запросит подтверждение удаления точки входа.
8. Нажмите на кнопку **Удалить**. Точка входа будет удалена с конфигурации конвейера.

Настройка точки входа

Набор параметров точки входа отличается в зависимости от ее типа. В системе доступны следующие типы точек входа:

- AMQP
- Audit
- Database
- Demo Logs
- HTTP Client
- HTTP Server
- Kafka
- Logstash
- Netflow
- R-Vision Endpoint
- Socket

- Syslog
- Vector
- WMI
- NATS JetStream
- Global Bus
- R-Vision EVO Endpoint

AMQP

При создании точки входа типа AMQP требуется указать значения следующих параметров:

1. **Адрес подключения**: выбор секрета типа **Строка подключения**, хранящего URI-адрес сервера AMQP в формате: `amqp://<user>:<password>@<host>:<port>/<vhost>?timeout=<seconds>`.

   Например: `amqp://user:password@127.0.0.1:5672/%2f?timeout=10`.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

2. **Идентификатор потребителя** (опционально): идентификатор группы потребителей, который будет использоваться для получения событий.

   Например: `event-consumer-group`.

3. **Ключ exchange** (опционально): ввод ключа exchange.
4. **Ключ смещения AMQP** (опционально): ввод ключа смещения AMQP.
5. **Название очереди** (опционально): ввод названия очереди.
6. **Ключ маршрутизации AMQP** (опционально): ввод ключа маршрутизации AMQP.
7. **Разделитель** (опционально): выбор способа разделения событий. Варианты: Символ, Новая строка, Подсчет октетов (RFC 6587).
8. **Настройки разделителя**: зависят от выбранного типа разделителя. Включают в себя настройку символа разделителя и максимальной длины события.

   Максимальная длина (опционально): максимальная длина события, зависит от выбранного разделителя. Должно быть целым числом без знака, в диапазоне от 0 до 4294967295.

9. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Audit

Точка входа Audit собирает события сервиса аудита и доставляет их на конвейер для дальнейшей обработки и анализа.

Сервис аудита предназначен для сбора, анализа и хранения данных о действиях пользователей и событиях в системе. Сервис аудита позволяет фиксировать такие действия, как входы в систему, создание, удаление и изменение ресурсов системы и конфигурации ее отдельных компонентов.

С помощью точки входа Audit можно отслеживать действия пользователей в системе для предотвращения, обнаружения и реагирования на возможные угрозы или нарушения безопасности. Например, вы можете настроить конфигурацию конвейера на выполнение следующих операций:

- Отслеживание действий суперадминистраторов
- Отслеживание действий над объектами

При создании точки входа типа Audit требуется указать значения параметра **Уровни угрозы**. Доступные варианты:

- **Отладочный** — уровень угрозы 1 (`DEBUG`);
- **Низкий** — уровень угрозы 2 (`LOW`);
- **Средний** — уровень угрозы 3 (`MEDIUM`);
- **Высокий** — уровень угрозы 4 (`HIGH`);
- **Критический** — уровень угрозы 5 (`CRITICAL`);
- **Фатальный** — уровень угрозы 6 (`FATAL`);
- **Аварийный** — уровень угрозы 7 (`ACCIDENT`).

Параметр фильтрует поток входящих событий аудита системы по их уровню угрозы.

Database

Каждая точка входа типа Database, размещенная на конвейере коллектора, использует 256 МБ оперативной памяти (RAM) из ресурсов коллектора. При размещении точек входа типа Database, необходимо учитывать их суммарное потребление памяти при распределении ресурсов коллектора.

При создании точки входа типа Database требуется указать значения следующих параметров:

1. **Драйвер базы данных**: выбор драйвера БД. Доступны драйверы следующих баз данных:
    - ClickHouse;
    - PostgreSQL;
    - MS SQL;
    - MySQL;

- Oracle DB.

2. **Адрес подключения**: выбор секрета типа **Строка подключения**, хранящего URI-адрес для подключения к базе данных. Формат URI должен соответствовать требованиям выбранной базы данных:

   - **PostgreSQL**: `jdbc:postgresql://[host][:<port>]/[database][?user=<user>&password=<password>&<parameters>]`

     Здесь:

     - `host` — адрес подключения к БД в формате `<хост>[:<порт>]`. Например: `localhost:5433`.
     - `database` — имя БД.
     - `user` — учетные данные пользователя БД.
     - `password` — пароль пользователя БД.
     - `parameters` — дополнительные параметры для подключения к БД в формате `<параметр>=<значение>`. Параметры отделяются друг от друга символом `&`.

       Учетные данные, адрес подключения и имя БД можно задавать в виде параметров.

     Полный список параметров приведен в документации PostgreSQL.

     Примеры строки подключения:

     - `jdbc:postgresql://localhost/testdb?user=user&password=secret&ssl=true`
     - `jdbc:postgresql://localhost:5432/testdb?user=user&password=password`

   - **ClickHouse**: `jdbc:clickhouse://[user@][host][/database][?parameters]`

     Здесь:

     - `user` — учетные данные пользователя БД в формате `<имя_пользователя>[:<пароль>]`. Например: `user:password`.
     - `host` — адрес подключения к БД в формате `<хост>[:<порт>]`. Например: `localhost:5433`.
     - `database` — имя БД.
     - `parameters` — дополнительные параметры для подключения к БД в формате `<параметр>=<значение>`. Параметры отделяются друг от друга символом `&`.

       Учетные данные, адрес подключения и имя БД можно задавать в виде параметров.

     Параметры ClickHouse идентичны параметрам PostgreSQL. Полный список параметров приведен в документации PostgreSQL.

     При работе с БД ClickHouse в качестве точки входа следует соблюдать следующие рекомендации:

- Не следует обращаться к базе данных `system`, поскольку она недоступна для чтения.
- Рекомендуется использовать сжатие передаваемых данных алгоритмом `gzip`. Для этого используйте параметр `compress_algorithm=gzip` в строке подключения.

Примеры строки подключения:

- `jdbc:clickhouse://user:password@localhost/database&compress_algorithm=gzip`
- `jdbc:clickhouse://localhost/database?user=user&password=password&compress_algorithm=gzip`

- **MS SQL**: `jdbc:sqlserver://[host][;parameters]`

Здесь:

- `host` — адрес подключения к БД в формате `<хост>[\<имя_экземпляра>][:<порт>]`. Например: `localhost\instance1:1433`.
- `parameters` — дополнительные параметры для подключения к БД в формате `<параметр>=<значение>`. Параметры отделяются друг от друга символом `;`.

Учетные данные пользователя и имя БД можно задавать в виде параметров.

Полный список параметров приведен в документации драйвера JDBC для SQL Server.

Примеры строки подключения:

- `jdbc:sqlserver://localhost;user=user;password=password`
- `jdbc:sqlserver://localhost:1433;database=database`

- **MySQL**: `jdbc:mysql://[user@][host][/database][?parameters]`

Здесь:

- `user` — учетные данные пользователя БД в формате `<имя_пользователя>[:<пароль>]`. Например: `user:password`.
- `host` — адрес подключения к БД в формате `<хост>[:<порт>]`. Например: `localhost:3306`.
- `database` — имя БД.
- `parameters` — дополнительные параметры для подключения к БД в формате `<параметр>=<значение>`. Параметры отделяются друг от друга символом `&`.

Учетные данные, адрес подключения и имя БД можно задавать в виде параметров.

Полный список параметров приведен в документации MySQL.

Примеры строки подключения:

- `jdbc:mysql://user:password@localhost:3306/database`
- `jdbc:mysql://localhost:3306/database?user=user&password=password`

◦ **Oracle** **DB**: `jdbc:oracle:thin:@[[protocol:]//]host[/service][:server_mode][/instance][?parameters]`

Здесь:

- `protocol` — протокол передачи данных. Например: `tcp`.
- `host` — адрес подключения к БД в формате `<хост>:[<порт>]`. Например: `localhost:1521`.
- `service` — имя сервиса для подключения к экземпляру БД.
- `server_mode` — режим работы сервера БД. Доступные значения:
  - `dedicated` — выделенный сервер;
  - `shared` — общий сервер;
  - `pooled` — сервер, добавленный в пул.
- `instance` — имя экземпляра БД.
- `parameters` — дополнительные параметры для подключения к БД в формате `<параметр>=<значение>`. Параметры отделяются друг от друга символом `&`.

Полный список параметров приведен в документации Oracle.

Примеры строки подключения:

- `jdbc:oracle:thin:@tcp//localhost:1521/service`
- `jdbc:oracle:thin:@tcp//localhost:1521/service/instance1`

Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

3. **SQL запрос**: ввод SQL запроса для работы с базой данных.

Пример 35. Пример SQL запроса

```
SELECT id, raw, collectorId FROM `default`.EventStorage_1072 limit 10;
```

При написании SQL-запросов, которые объединяют несколько таблиц, избегайте одинаковых имен столбцов, чтобы предотвратить ошибки. Для этого:

◦ Вместо `SELECT *` перечисляйте конкретные столбцы.

◦ Используйте псевдонимы для столбцов с одинаковыми именами.

Например, вместо запроса вида

```
SELECT
    *
FROM
    Database1.TableStart AS tableStart
```

```
JOIN Database1.TableStop AS tableStop
ON
    tableStart.id = tableStop.id
WHERE
    tableStart.id > ?;
```

используйте запрос

```
SELECT
    tableStart.id AS tableStartId,
    /* псевдоним для столбца id в первой таблице */
    tableStart.AdminId,
    tableStart.StartTime,
    tableStop.id,
    tableStop.EndTime,
    tableStop.IsSuccessful
    /* перечислены конкретные столбцы */
FROM
    Database1.TableStart AS tableStart
JOIN Database1.TableStop AS tableStop
ON
    tableStart.id = tableStop.id
WHERE
    tableStart.id > ?;
```

4. **Поле идентификатора**: добавление полей-идентификаторов для определения строки начала чтения при первом подключении к БД. Список полей представлен в виде таблицы со столбцом для указания ключа поля идентификатора и столбцом для задания его стартового значения.

Система хранит значение последней обработанной записи и при следующем выполнении запроса модифицирует условие выборки, инициируя чтение данных не с указанной строки в **Поле идентификатора**, а с записи, следующей непосредственно за последней обработанной.

Необходимо гарантировать, что значения всех полей, определенных в качестве идентификаторов в строке строго согласуются между собой, отражая уникальные данные одной записи в базе данных.

5. **Интервал запроса, секунд** (опционально): интервал между запросами в секундах.

Интервал между инициациями SQL-запросов не охватывает время исполнения запросов, что может фактически продлить интервал между запусками.

6. **Максимальное количество строк** (опционально): максимальное количество строк для чтения в ходе выполнения SQL-запроса.

Пример настройки точки входа Database

Представим, что в подключаемой базе данных имеется таблица событий `events`, из которой необходимо извлечь данные по столбцам `id`, `timestamp` и `data`.

После определения типа базы данных и конфигурации строки подключения, следующий шаг заключается в настройке идентификационных полей для отслеживания и выборки данных. В качестве таких полей можно использовать `id` и `timestamp`. Добавим их ключи в левый столбец таблицы **Поле идентификатора**.

Необходимо гарантировать, что значения всех полей, определенных в качестве идентификаторов в строке — таких как `id`, `timestamp` и любые другие участвующие поля — строго согласуются между собой, отражая уникальные данные одной записи в базе данных. Эти поля в совокупности формируют составной ключ, обеспечивая точную идентификацию записи в БД.

Предположим, что первые 50 записей не должны попасть в выборку. Учитывая, что идентификаторы (`id`) в базе данных присваиваются последовательно, установим для поля `id` в **Поле идентификатора** значение "51". Таким образом, извлечение данных начнется непосредственно с записи, имеющей `id=51`.

Если предположить, что запись с `id=51` соответствует событию, произошедшему в "2024-01-01 00:00:00", это значение также указывается в качестве критерия выборки в **Поле идентификатора** для `timestamp`. Использование `id` в сочетании с `timestamp` в качестве начальных условий выборки формирует составной ключ, обеспечивающий уникальность каждой записи в базе данных.

Определение и использование составных ключей, основанных на нескольких идентификаторах, таких как `id` и `timestamp`, повышает точность выборки данных, минимизируя возможность дублирования или пропуска записей при изменении условий выборки или в случае неоднозначности идентификации записей только по одному критерию.

В целях снижения нагрузки на коллектор можно ограничить считывание данных из БД порциями по 100 записей, используя в SQL-запросе оператор `LIMIT`. Например:

```
SELECT id, timestamp, data FROM events WHERE id > ? ORDER BY id ASC LIMIT 100;
```

Здесь `?` — это параметр, который позволяет динамически подставлять значение идентификатора (`id`) последней обработанной записи в базе данных. Система хранит `id` последней обработанной записи и при последующем выполнении запроса адаптирует условие выборки, запуская процесс чтения данных не с фиксированной строки (в данном случае, не с 51), а с записи, следующей непосредственно за последней обработанной.

Предположим, новые данные поступают в рассматриваемую базу данных с периодичностью в 60 минут. Для оптимизации процесса извлечения данных рекомендуется настроить периодичность выполнения SQL-запросов, соответствующую этому интервалу. Оптимальным решением будет установка интервала в 3600 секунд (один час), что позволит системе регулярно обновлять данные без излишних запросов.

Указанный интервал между инициациями SQL-запросов не охватывает время

исполнения запросов, что может фактически продлить интервал между запусками.

Таким образом, вышеприведенные настройки позволят считать из таблицы `events` в подключенной БД все события начиная с `id=51` и `timestamp=`2024-01-01 00:00:00`` порциями по 100 записей. Порции записей будут направляться в точку входа раз в 3600 секунд.

Demo Logs

Точка входа Demo Logs предназначена для генерации случайных событий в целях демонстрации или тестирования. Формат, в котором будут генерироваться события, определяется пользователем.

При создании точки входа типа Demo Logs требуется указать значения следующих параметров:

1. **Количество событий** (опционально): количество событий для вывода. По умолчанию точка входа выводит логи непрерывно. Формат: целое число больше 0.
2. **Формат генерируемых событий**: выбор формата, в котором будут генерироваться события. Варианты:
   - **apache_common** — общий журнал Apache;
   - **apache_error** — журнал ошибок Apache;
   - **bsd_syslog** — формат BSD syslog (RFC 3164);
   - **json** — логи HTTP-сервера в формате JSON;
   - **shuffle** — выбор событий из заданного списка;
   - **syslog** — формат syslog (RFC 5424).
3. **Интервал в секундах** (опционально): время задержки между пакетами событий вывода. Формат ввода: число с точностью до трех знаков после запятой, начиная с 0. По умолчанию события отправляются каждую секунду.
4. **Список событий**: отображается, если выбран формат shuffle.

   Чтобы добавить событие, нажмите на кнопку **Добавить**. Отобразится строка добавления события.

   Чтобы удалить событие, нажмите на кнопку в строке события.

   Пример 36. Примеры событий

```
{"dvendor":"Microsoft","dhost":"host1","duser":"user1","sourceIp":"192.0.2.0"}

{"dvendor":"Linux","dhost":"host1","duser":"user1","sourceIp":"198.51.100.0"}
```

Экранировать символы в списке событий не требуется.

При выборе формата shuffle события передаются в виде строки. Чтобы обрабатывать эти события, преобразуйте их в нужный формат, например, с помощью функции `.raw = parse_json!(.raw)`. Чтобы привести метку времени к ожидаемому типу и формату данных (`2024-07-15T09:05:25.153815269Z`), используйте

функцию parse_timestamp:

```
.timestamp = parse_timestamp!(.timestamp, format: "%FT%T%.9fZ")
```

HTTP Client

Точка входа HTTP Client выполняет роль клиента во взаимодействии типа "клиент — сервер". Она позволяет подключаться к внешнему серверу по HTTP-протоколу для активного сбора данных с этого сервера. Сбор данных происходит за счет регулярного выполнения HTTP-запросов, посылаемых от точки входа (клиента) к серверу.

При создании точки входа типа HTTP Client требуется указать значения следующих параметров:

1. **Адрес подключения**: полный адрес для подключения к HTTP-серверу в виде URI, например `http://127.0.0.1:9898/logs` или `http://somehost:9898/logs`.
2. **Тип аутентификации** (опционально): выбор типа аутентификации на HTTP-сервере:
   - **Базовая**: выберите секрет типа **Учетные данные**, хранящий имя пользователя и пароль для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   - **Токен**: выберите секрет типа **Токен**, хранящий токен для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

3. **HTTP-заголовки** (опционально): заголовки HTTP-запроса. Заголовки позволяют уточнить HTTP-запрос и определяют, какие данные требуется передавать в запросе.

   ▼ *Об HTTP-заголовках*

   HTTP-заголовок — это строка в HTTP-запросе, состоящая из пары "имя: значение". Он используется для уточнения запроса или описания содержимого сообщения. Формат заголовка определяется стандартом RFC 822.

   HTTP-заголовки делятся на следующие категории:

   - Основные заголовки (General Headers) — применяются во всех сообщениях клиента и сервера.
   - Заголовки запроса (Request Headers) — используются только в запросах клиента.
   - Заголовки ответа (Response Headers) — используются только в ответах от сервера.
   - Заголовки сущности (Entity Headers) — применяются к содержимому сообщения.

   Полный список HTTP-заголовков можно найти в MDN Web Docs.

Пример HTTP-заголовка

Для считывания данных с сервера порциями по 512 байт, в HTTP-запросе необходимо указать заголовок `Content-Length`. Значение этого заголовка

должно соответствовать максимальному размеру поступающих данных в байтах, в данном случае это будет `512`.

Чтобы задать заголовок, нажмите на кнопку **Добавить**. Отобразится строка добавления заголовка. Введите заголовок и его значение. Для одного заголовка можно задать несколько значений, разделенных запятой.

Чтобы удалить заголовок, нажмите на кнопку в строке нужного заголовка.

4. **Параметры запроса** (опционально): дополнительные параметры, применяемые к строке HTTP-запроса. Параметры представляют собой пары "ключ=значение" и позволяют конкретизировать ресурс, получаемый из HTTP-запроса.

Пример HTTP-параметра

Допустим, HTTP-ресурс содержит множество событий, из которых необходимо выбрать только события с уровнем угрозы **Критический**. Для указания критического уровня угрозы используется пара `severity=critical`. Таким образом, чтобы получить нужные данные, в HTTP-запросе необходимо передать параметр `severity` со значением `critical`.

Чтобы задать параметр, нажмите на кнопку **Добавить**. Отобразится строка добавления параметра. Введите параметр и его значение. Для одного параметра можно задать несколько значений, разделенных запятой.

Чтобы удалить параметр, нажмите на кнопку в строке нужного параметра.

5. **Интервал запроса, секунд** (опционально): интервал между HTTP-запросами в секундах. Интервал включает в себя время выполнения HTTP-запроса.

Интервал запроса рекомендуется устанавливать выше значения таймаута запроса.

6. **Таймаут запроса, секунд** (опционально): максимальное время выполнения HTTP-запроса в секундах. Если запрос будет выполняться дольше указанного времени, его выполнение будет прекращено.
7. **Адрес HTTP-прокси** (опционально): полный адрес для подключения к HTTP-прокси в виде URI. Применяется, если HTTP-сервер использует прокси.
8. **Хосты без прокси** (опционально): список хостов, для которых не используется прокси-сервер. Хосты указываются через запятую.

   Допускается использование доменных имен (например: example.com), шаблонов доменов (например: .example.com), IP-адресов (например: 127.0.0.1), CIDR-блоков (например: 192.168.0.0/16).

9. **Метод запроса** (опционально): выбор метода HTTP-запроса:
   - **DELETE** — удаляет данные с сервера.
   - **GET** — запрашивает данные с сервера. Метод предназначен только для получения данных без дополнительных взаимодействий с сервером.

- **HEAD** — запрашивает данные с сервера без тела HTTP-ответа.
- **PATCH** — используется для частичного изменения ресурса.
- **POST** — отправляет данные на сервер. Метод используется в тех случаях, когда для получения данных требуется дополнительное взаимодействие с сервером (например, передача данных для авторизации на сервере).
- **PUT** — заменяет данные на сервере данными из HTTP-запроса.

10. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

    a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

    Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

    b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

    Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Пример настройки точки входа HTTP Client

Представим, что требуется получить события с внешнего сервера. После определения URI-адреса подключения и типа аутентификации, следующий шаг заключается в настройке заголовков и параметров в отправляемых HTTP-запросах.

Для снижения нагрузки на сервер можно ограничить считывание данных порциями по 512 байт. Для этого добавим заголовок `Content-Length`, определяющий максимальный размер поступающих данных в байтах, и зададим ему соответствующее значение.

Предположим, что на сервере хранятся события с различным уровнем угрозы, которые определяются параметром `severity`. Чтобы получить только события с уровнем угрозы **Критический** (`critical`), добавим параметр `severity` с соответствующим значением.

Так как для доступа к серверу требуется передать данные для авторизации, для повышения безопасности их передачи следует использовать метод POST. В отличие от метода GET, POST кодирует данные и передает их в теле HTTP-запроса, обеспечивая их конфиденциальность.

Предположим, новые данные поступают на сервер с периодичностью 60 минут. Для оптимизации процесса извлечения данных рекомендуется настроить выполнение HTTP-запросов с таким же интервалом. Оптимальным будет установка интервала в 3600 секунд (один час), что позволит системе регулярно обновлять данные без излишних запросов.

Указанный интервал между HTTP-запросами включает время их выполнения.

Поскольку время выполнения HTTP-запросов может превысить указанный интервал, необходимо задать таймаут, ограничивающий максимальное время выполнения запросов. Оптимальным решением будет установка таймаута в 3000 секунд (50 минут), что гарантирует завершение текущего

запроса до инициации следующего.

Таким образом, вышеприведенные настройки позволят считывать с сервера события с критическим уровнем угрозы порциями по 512 байт. Порции записей будут направляться в точку входа каждые 3600 секунд.

HTTP Server

Точка входа HTTP Server выполняет роль сервера во взаимодействии типа "клиент — сервер". Она позволяет внешним источникам (клиентам) подключаться к ней для пассивного сбора данных с помощью HTTP-запросов. Обработка HTTP-запросов от клиентов приводит к генерации событий, которые поступают в точку входа (на сервер).

При создании точки входа типа HTTP Server требуется указать значения следующих параметров:

1. **Кодировка**: выбор формата кодировки событий. Варианты: Text (по умолчанию), JSON, Newline-delimited JSON, Binary.
2. **Путь** (опционально): URL-путь, используемый клиентом для записи событий при использовании метода POST. Например: `/logs` или `/logs/application`.

   Путь будет добавлен в метаданные поступающих событий.

   По умолчанию точка входа обрабатывает HTTP-запросы, поступающие по любому URL-пути, соответствующему заданному шаблону. Например, при настройке пути в виде `/logs`, система также обработает запросы, поступающие по вложенным путям, таким как `/logs/application`. Чтобы разрешить получение HTTP-запросов только с указанного пути, установите флажок **Абсолютный путь**. В этом случае все события, поступающие на путь, отличный от заданного, будут отбрасываться.

3. **HTTP-заголовки** (опционально): заголовки поступающих HTTP-запросов. Заголовки позволяют уточнить, какие дополнительные данные требуется получить от клиентов. Указанные заголовки будут добавлены в метаданные поступающих событий. Допустимо использование подстановочного символа *, например, `X-*`.

   ▼ *Об HTTP-заголовках*

   HTTP-заголовок — это строка в HTTP-запросе, состоящая из пары "имя: значение". Он используется для уточнения запроса или описания содержимого сообщения. Формат заголовка определяется стандартом RFC 822.

   HTTP-заголовки делятся на следующие категории:

   - Основные заголовки (General Headers) — применяются во всех сообщениях клиента и сервера.
   - Заголовки запроса (Request Headers) — используются только в запросах клиента.
   - Заголовки ответа (Response Headers) — используются только в ответах от сервера.
   - Заголовки сущности (Entity Headers) — применяются к содержимому сообщения.

   Полный список HTTP-заголовков можно найти в MDN Web Docs.

   Пример

> Чтобы определить, от какого агента поступило событие, необходимо указать заголовок `User-Agent`.

Чтобы задать заголовок, нажмите на кнопку **Добавить**. Отобразится строка добавления заголовка. Введите заголовок.

+ Чтобы удалить заголовок, нажмите на кнопку в строке нужного заголовка.

4. **Параметры HTTP-запроса** (опционально): дополнительные параметры, применяемые к строке поступающих HTTP-запросов. Как и HTTP-заголовки, параметры позволяют конкретизировать данные, получаемые от клиентов. Указанные параметры будут добавлены в метаданные поступающих событий. Допустимо использование подстановочного символа *, например, `X-*`.

   Пример

   Предположим, что от клиентов поступают события с различным уровнем угрозы. Чтобы определить уровень угрозы каждого события, необходимо указать соответствующий параметр, например, `severity`.

   Чтобы задать параметр, нажмите на кнопку **Добавить**. Отобразится строка добавления параметра. Введите параметр.

   Чтобы удалить параметр, нажмите на кнопку в строке нужного параметра.

5. **Метод запроса** (опционально): выбор метода поступающих HTTP-запросов:
   - **DELETE** — удаляет данные с сервера.
   - **GET** — запрашивает данные с сервера с полным HTTP-ответом.
   - **HEAD** — запрашивает данные с сервера без тела HTTP-ответа.
   - **PATCH** — используется для частичного изменения ресурса.
   - **POST** — отправляет данные на сервер.
   - **PUT** — заменяет данные на сервере данными из HTTP-запроса.
6. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

7. **Порт точки входа**: номер порта для обмена данными между точкой входа и клиентом. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.

Пример настройки точки входа HTTP Server

Предположим, требуется получать события от внешних клиентов. После определения порта точки входа, кодировки событий и пути записи событий, следующим шагом будет настройка заголовков и параметров в поступающих HTTP-запросах.

Допустим, необходимо определить, от какого агента поступило каждое событие. Для этого добавим заголовок `User-Agent`. В метаданные событий будет добавлено одноименное поле с информацией об агенте-источнике.

Предположим, что от клиентов поступают события с различным уровнем угрозы, которые определяются параметром `severity`. Чтобы определить уровень угрозы каждого события, добавим этот параметр. В метаданные событий будет добавлено одноименное поле с информацией об уровне угрозы.

Так как в данном примере предполагается только получение событий, в качестве метода поступающих HTTP-запросов выберем GET.

Таким образом, эти настройки позволят получать от клиентов события с дополнительной информацией об агентах-источниках и уровнях угрозы.

Kafka

При создании точки входа типа Kafka требуется указать значения следующих параметров:

1. **Адреса подключения**: список адресов серверов в кластере Kafka для подключения. Формат: IP-адрес:порт, например, 10.14.22.123:9092, 10.14.23.332:9092.

   Чтобы добавить адрес, нажмите на кнопку **Добавить**. Отобразится строка добавления адреса.

   Чтобы удалить адрес, нажмите на кнопку в строке адреса.

2. **Название группы потребителей**: название Consumer group, которое будет использоваться для получения событий из Kafka, например, event-consumer-group.
3. **Названия топиков**: названия топиков Kafka, из которых будут читаться события.

   Чтобы добавить топик, нажмите на кнопку **Добавить**. Отобразится строка добавления топика.

   Чтобы удалить топик, нажмите на кнопку в строке с названием топика.

4. **Разделитель** (опционально): выбор способа разделения событий. Варианты: Символ, Новая строка, Подсчет октетов (RFC 6587).
5. **Настройки разделителя**: зависят от выбранного типа разделителя. Включают настройку символа разделителя и максимальной длины события.
6. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Logstash

При создании точки входа типа Logstash требуется указать значения следующих параметров:

1. **Порт точки входа**: номер порта для обмена данными между точкой входа и внешней системой. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.
2. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

3. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Netflow

Точка входа поддерживает следующие версии протокола Netflow: v5, v7, v9, IPFIX.

При создании точки входа типа Netflow требуется указать значения следующих параметров:

1. **Порт точки входа**: номер порта для обмена данными между точкой входа и внешней системой. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.
2. **Размер буфера** (опционально): определение максимального размера буфера для входящих сообщений, в байтах.

R-Vision Endpoint

При создании точки входа типа R-Vision Endpoint требуется указать значения следующих параметров:

1. **Порт точки входа**: номер порта для обмена данными между точкой входа и R-Vision Endpoint. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.
2. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Socket

При создании точки входа типа Socket (TCP, UDP) требуется указать значения следующих параметров:

1. **Порт точки входа**: номер порта для обмена данными между точкой входа и внешней системой. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.
2. **Протокол**: выбор протокола передачи данных. Варианты: TCP, UDP.
3. **Разделитель** (опционально): выбор способа разделения событий. Варианты: Символ, Новая строка, Подсчет октетов (RFC 6587).
4. **Настройки разделителя**: зависят от выбранного типа разделителя. Включают в себя настройку символа разделителя и максимальной длины события.
5. **Размер буфера** (опционально): определение максимального размера буфера для входящих сообщений, в байтах.

   Настройка размера буфера доступна только при выборе протокола UDP.

6. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

      Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку

Открыть.

b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Syslog

При создании точки входа типа Syslog требуется указать значения следующих параметров:

1. **Порт точки входа**: номер порта для обмена данными между точкой входа и внешней системой. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.
2. **Протокол**: выбор протокола передачи данных. Варианты: TCP, UDP.
3. **Размер буфера** (опционально): определение максимального размера буфера для входящих сообщений, в байтах.
4. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Vector

При создании точки входа типа Vector требуется указать значения следующих параметров:

1. **Порт точки входа**: номер порта для обмена данными между точкой входа и системой. Допустимый диапазон: 30000—32767. Значение должно быть уникальным в рамках кластера.
2. **Включить TLS** (опционально): использование шифрования TLS. При включении TLS необходимо:

   a. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   b. Выбрать секрет типа **Сертификат**, хранящий данные корневого сертификата.

Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

WMI

Точка входа типа WMI (Windows Management Instrumentation) предназначена для получения данных с удаленных хостов на базе ОС Windows. Точка входа собирает события журналов и другие системные данные через WMI-запросы, обеспечивая доступ к информации о процессах, службах и других ресурсах системы.

При создании точки входа WMI требуется указать значения следующих параметров:

1. **Адрес подключения**: имя хоста или IP-адрес удаленного хоста, с которого будут собираться данные, например, localhost, 198.51.100.123.

   Не рекомендуется создавать в одном коллекторе несколько точек WMI, подключенных к одному адресу.

   Если необходимо получать данные из нескольких журналов, выберите их все в одной точке.

2. **Учетные данные**: выбор учетной записи, которая будет использоваться для подключения к удаленному хосту и доступа к журналам. Необходимо, чтобы учетная запись имела права на доступ к журналам безопасности на целевом хосте.

   Учетные данные выбираются из списка секретов.

3. **Журнал**: выбор журналов, из которых необходимо собирать события.

   Доступны следующие журналы:

   - **Application** — журнал приложений.
   - **Security** — журнал безопасности.
   - **System** — журнал системы.
   - **Setup** — журнал установки.
   - **Forwarded Events** — журнал событий передовой диагностики.
   - **Hardware Events** — журнал аппаратного обеспечения.
   - **Microsoft-Windows-Sysmon/Operational** — журнал операций System Monitor.
   - **Microsoft-Windows-PowerShell/Operational** — журнал операций PowerShell.
   - **Microsoft-Windows-Bits-Client/Operational** — журнал службы Background Intelligent Transfer Service.
   - **Microsoft-Windows-DNSServer/Audit** — журнал аудита DNS-запросов.
   - **Microsoft-Windows-TaskScheduler/Operational** — журнал операций Планировщика задач.
   - **Microsoft-Windows-WMI-Activity/Operational** — журнал активности WMI.

- **Microsoft-Windows-WinRM/Operational** — журнал операций Windows Remote Management.
- **Microsoft-Windows-AppLocker/EXE and DLL** — журнал применения политик AppLocker для EXE и DLL.
- **DNS Server** — журнал DNS-сервера.
- **Microsoft-Windows-DHCP Server Events/Operational** — журнал аудита конфигурации DHCP.
- **Microsoft-Windows-DHCP Server Events/FilterNotifications** — журнал событий фильтрации DHCP-сервера.

Название журнала, из которого было получено событие, сохраняется в поле `@wmi.channel` в метаданных события.

4. **Интервал запроса, секунд**: интервал между запросами, включая время выполнения запроса. По умолчанию установлено значение 60 секунд.

Интервал между инициациями запросов охватывает время исполнения запросов, что может фактически продлить интервал между запусками.

5. **Кодировка** (опционально): кодировка данных, полученных от источника. Позволяет задать кодировку, если требуется специальная обработка текстовых данных. Формат: название кодировки, например, UTF-8, Windows-1251.

NATS JetStream

Точка входа типа NATS JetStream предназначена для получения данных с внешних систем через брокер сообщений NATS.

При создании точки входа типа NATS JetStream требуется указать значения следующих параметров:

1. **Адрес подключения**: полный путь для подключения к NATS в формате URL- или IP-адреса с указанием порта. Например: `nats://cluster.nats.com` или `nats://127.0.0.1:4242`.

Для обмена сообщениями через NATS по умолчанию используется порт 4222.

2. **Название потока** (опционально): название существующего потока (хранилища сообщений), из которого необходимо выполнять чтение тем.

Поток должен быть создан в NATS до настройки точки входа.

3. **Тема**: список тем NATS, из которых необходимо получать сообщения.

Темы должны быть созданы в NATS до настройки точки входа.

Чтобы добавить тему, нажмите на кнопку **Добавить**. Отобразится строка добавления темы.

Чтобы удалить тему, нажмите на кнопку в строке темы.

Особенности ввода тем:

- Название темы вводится в кодировке Unicode.
- Для задания иерархии и обозначения связанных тем можно использовать точку. Например: `time.us.east`.
- Допустимо использование подстановочных символов:
  - `*` — единственный токен. Например: `time.*.east`.
  - `>` — множественные токены. Данный символ может использоваться только в конечной позиции. Например: `time.us.>`.

4. **Домен** (опционально): название домена NATS JetStream.
5. **Ключ темы** (опционально): ключ, под которым будет записана тема NATS.
6. **Потребитель**: название потребителя, который будет получать сообщения.
7. **Название соединения** (опционально): название соединения с NATS.
8. **Лимит сообщений** (опционально): максимальное количество сообщений, которые могут храниться в буфере. Буфер используется для накопления сообщений, когда они не могут быть направлены в точку входа (например, когда сообщения поступают быстрее, чем точка входа может их обработать). Сообщения находятся в буфере до тех пор, пока их не удастся успешно обработать точкой входа. При достижении лимита новые сообщения не будут сохраняться в буфере.
9. **Тип аутентификации** (опционально): выбор типа аутентификации во внешней системе:
10. **Базовая**: выберите секрет типа **Учетные данные**, хранящий имя пользователя и пароль для авторизации.

    Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

11. **Токен**: выберите секрет типа **Токен**, хранящий токен для авторизации.

    Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

12. **NKey**: выберите секрет типа **NATS Key**, хранящий приватный и публичный ключи для подключения к NATS.

    Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

13. **Формат событий**: формат получения событий из внешней системы. Доступные форматы:
    - **AVRO**. При выборе этого формата требуется ввести схему AVRO в формате JSON. Например:

      ```
      { "type": "record", "name": "log", "fields": [{ "name": "msg", "type": "string"
      }] }
      ```

    - **BYTES**. Данный формат выбран по умолчанию.
    - **GELF**.

- **INFLUXDB**.
- **JSON**.
- **NATIVE**.
- **NATIVE_JSON**.
- **PROTOBUF**. При выборе этого формата требуется загрузить файл с описанием протокола Protocol Buffers (Protobuf). Подробную информацию о структуре файла можно найти в официальной документации Protocol Buffers.
- **SYSLOG**.
- **VRL**. При выборе этого формата требуется указать код на языке VRL для обработки каждого поступающего события, а также ввести имя часового пояса для преобразования временных меток, которые не содержат часовой пояс в явном виде.

Global Bus

Точка входа типа Global Bus предназначена для получения событий из глобальной шины.

Данная точка входа необходима для работы с коллекторами вне центрального пространства, но также может использоваться и с коллекторами в центральном пространстве.

Необходимость использования глобальной шины для отправки и получения событий из вынесенных коллекторов обусловлена тем, что запись событий в хранилище доступна только на конвейере коллектора, находящегося в центральном пространстве.

При создании точки входа типа Global Bus требуется выбрать из выпадающего списка коллектор, из которого необходимо получать события.

R-Vision EVO Endpoint

Точка входа типа R-Vision EVO Endpoint предназначена для получения событий с агентов, установленных на конечных устройствах.

При создании точки входа типа R-Vision EVO Endpoint требуется указать домен NATS JetStream для подключения к шлюзу. Домен задается в конфигурации NATS-сервера в формате `gw-<gateway_id>`, где `<gateway_id>` — идентификатор шлюза. Подключение выполняется к NATS-серверу, заданному по умолчанию: `nats://nats-space:4222`.

Работа с VRL-трансформациями

Данный раздел описывает процесс работы с VRL-трансформациями на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О VRL-трансформациях

VRL-трансформация — это элемент конвейера, который позволяет обрабатывать и преобразовывать события с применением языка VRL.

На диаграмме конфигурации конвейера VRL-трансформация имеет один вход и один выход. Этот

элемент конвейера принимает на вход события, проводит их преобразование и на выходе предоставляет уже преобразованные события.

В отличие от нормализаторов, VRL-трансформация позволяет задать простые правила преобразования событий на языке VRL без предварительного создания правил нормализации и привязки к ним.

VRL-трансформация удобна в тех случаях, когда требуется провести несложные модификации в структуре событий: например, использовать одну или несколько функций для упрощения дальнейшей работы с событиями или добавить/изменить/удалить какое-нибудь поле в структуре событий.

Для проведения более сложных преобразований, применения одних и тех же преобразований в нескольких конвейерах, поддержки версионирования (возможности изменения правил преобразований без перезапуска конвейера) рекомендуется использовать нормализаторы, основанные на правилах нормализации.

Доступные операции над VRL-трансформацией:

- Добавление VRL-трансформации
- Изменение VRL-трансформации
- Удаление VRL-трансформации

Добавление VRL-трансформации

Чтобы добавить VRL-трансформацию на конфигурацию конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **VRL-трансформация**. Отобразится окно добавления VRL-трансформации.
8. При необходимости выберите шаблон для автоматического заполнения полей VRL-трансформации. По умолчанию выбран вариант **Без шаблона**.
9. Введите название VRL-трансформации.
10. Введите правило, по которому будет преобразовываться событие, на языке VRL.

В редакторе кода доступны горячие клавиши.

В VRL-программе для доступа к данным события используется символ точки (`.`), а к метаданным события — символ процента (`%`). В корреляционных событиях метаданные включают данные исходных событий.

11. При необходимости из выпадающего списка **Обработка ошибок** выберите шину коллектора, в которую будет отправлено событие в случае возникновения ошибок обработки. По умолчанию выбран вариант **События с ошибкой**.
12. Установите флажок **Сохранить как шаблон**, если настройку VRL-трансформации необходимо сохранить в качестве шаблона для дальнейшего использования.
13. Нажмите на кнопку **Добавить**. Новая VRL-трансформация отобразится на диаграмме конфигурации.

Если в добавленных правилах содержатся отсутствующие активные списки или таблицы обогащения, система отобразит список ошибок с указанием ID и названий правил.

Чтобы добавить конфигурацию с правилами, где обнаружены ошибки, установите флажок **Игнорировать ошибки**. Если флажок не установлен, правила с ошибками будут исключены.

Нажмите на кнопку **Готово**, чтобы сохранить конфигурацию.

Пример:

Условие на языке VRL, согласно которому исходное событие будет обогащаться дополнительным полем `responseStatus`. В нем будет устанавливаться значение "Bad request", если в поле `status` приходящего на VRL-трансформацию события установлено значение 401 или 404:

Пример 37. Пример VRL-трансформации, язык VRL

```
if .status == "404" || .status == "401" {
.responseStatus = "Bad request"
}
```

Изменение VRL-трансформации

Чтобы изменить VRL-трансформацию:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.

6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Откройте окно настроек VRL-трансформации одним из следующих способов:
   - Нажмите на кнопку действий (⋮) справа от названия VRL-трансформации и выберите опцию **Изменить**.
   - Дважды нажмите на VRL-трансформацию на диаграмме конфигурации конвейера.
8. Внесите изменения в требуемые поля.
9. Нажмите на кнопку **Сохранить**. Измененные данные VRL-трансформации будут сохранены.

Удаление VRL-трансформации

Чтобы удалить VRL-трансформацию:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку действий (⋮) справа от названия VRL-трансформации и выберите опцию **Удалить**. Система отобразит окно подтверждения удаления.
8. Нажмите на кнопку **Удалить**. VRL-трансформация будет удалена.

Работа с фильтрами

Данный раздел описывает процесс работы с фильтрами на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О фильтрах

Фильтр — это элемент конвейера, который позволяет сократить объем обрабатываемых событий путем их отсеивания на основе условий, заданных на языке VRL. На диаграмме конфигурации конвейера фильтр имеет один вход и один выход. На входе каждое событие проверяется по условию фильтра и появляется на выходе только в том случае, если условие выполняется. В противном случае событие отсеивается и не участвует в дальнейшей обработке.

Доступные операции над фильтром:

- Добавление фильтра
- Изменение фильтра

- Удаление фильтра

Добавление фильтра

Чтобы добавить фильтр на конфигурацию конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Фильтр**. Отобразится окно добавления фильтра.
8. При необходимости выберите шаблон для автоматического заполнения полей фильтра. По умолчанию выбран вариант **Без шаблона**.
9. Введите название фильтра.
10. Задайте условие фильтрации входящих событий на языке VRL.

    Условия обычно формулируются через имена полей и их значения, и для их определения используются математические и логические операторы.

    В редакторе кода доступны горячие клавиши.

    В VRL-программе для доступа к данным события используется символ точки (`.`), а к метаданным события — символ процента (`%`). В корреляционных событиях метаданные включают данные исходных событий.

11. Установите флажок **Сохранить как шаблон**, если настройку фильтра необходимо сохранить в качестве шаблона для дальнейшего использования.
12. Нажмите на кнопку **Добавить**. Новый фильтр отобразится на диаграмме конфигурации.

    Если в добавленных правилах содержатся отсутствующие активные списки или таблицы обогащения, система отобразит список ошибок с указанием ID и названий правил.

    Чтобы добавить конфигурацию с правилами, где обнаружены ошибки, установите флажок **Игнорировать ошибки**. Если флажок не установлен, правила с ошибками будут исключены.

    Нажмите на кнопку **Готово**, чтобы сохранить конфигурацию.

Пример:

Условие фильтра, пропускающего только события, поля `externalId` и `categoryOutcome` которых имеют значения "4648" и "Success" соответственно:

```
.externalId == "4648" && .categoryOutcome == "Success"
```

Изменение фильтра

Чтобы изменить фильтр:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Откройте окно настроек фильтра одним из следующих способов:
   - Нажмите на кнопку действий (⋮) справа от названия фильтра и выберите опцию **Изменить**.
   - Дважды нажмите на фильтр на диаграмме конфигурации конвейера.
8. Внесите изменения в требуемые поля.
9. Нажмите на кнопку **Сохранить**. Измененные данные фильтра будут сохранены.

Удаление фильтра

Чтобы удалить фильтр:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.

7. Нажмите на кнопку действий (⋮) справа от названия фильтра и выберите опцию **Удалить**. Система отобразит окно подтверждения удаления.
8. Нажмите на кнопку **Удалить**. Фильтр будет удален.

Работа с маршрутизаторами

Данный раздел описывает процесс работы с маршрутизаторами на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О маршрутизаторах

Маршрутизатор — это элемент конвейера, который позволяет на основе заданных фильтров (маршрутов) направлять события на разные этапы обработки в зависимости от их содержимого или других условий. На диаграмме конфигурации конвейера маршрутизатор имеет один вход и несколько выходов: по одному выходу на каждый настроенный в нем маршрут. На входе каждое событие проверяется по условиям маршрутов и появляется на выходах тех маршрутов, где условие выполняется.

Если событие не удовлетворяет условиям ни одного из маршрутов, то оно направляется на выход маршрутизатора **По умолчанию** (если установлен флажок **Добавить маршрут по умолчанию**) или удаляется (если флажок **Добавить маршрут по умолчанию** не установлен).

Доступные операции над маршрутизатором:

- Добавление маршрутизатора
- Изменение маршрутизатора
- Удаление маршрутизатора
- Настройка маршрутов

Добавление маршрутизатора

Чтобы добавить маршрутизатор на конфигурацию конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Маршрутизатор**. Отобразится окно добавления маршрутизатора.
8. Введите название маршрутизатора.

Для маршрутизатора недопустимо использование названий "_unmatched" и "_default".

9. Настройте список маршрутов маршрутизатора.
10. При необходимости установите флажок **Добавить маршрут по умолчанию**. В этом случае к маршрутизатору будет добавлен маршрут **По умолчанию**, на выход которого будут попадать те события, которые не удовлетворяют условиям остальных маршрутов.

    При неустановленном флажке события, не удовлетворяющие условиям ни одного из маршрутов, будут удаляться.

11. Нажмите на кнопку **Добавить**. Новый маршрутизатор отобразится на диаграмме конфигурации.

Изменение маршрутизатора

Чтобы изменить маршрутизатор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Откройте окно настроек маршрутизатора одним из следующих способов:
   - Нажмите на кнопку действий (⋮) справа от названия маршрутизатора и выберите опцию **Изменить**.
   - Дважды нажмите на маршрутизатор на диаграмме конфигурации конвейера.
8. Внесите изменения в требуемые поля.
9. Нажмите на кнопку **Сохранить**. Измененные данные маршрутизатора будут сохранены.

Удаление маршрутизатора

Чтобы удалить маршрутизатор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.

4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
8. Нажмите на кнопку действий (⋮) справа от названия маршрутизатора и выберите опцию **Удалить**. Система отобразит окно подтверждения удаления.
9. Нажмите на кнопку **Удалить**. Маршрутизатор будет удален.

Настройка маршрутов

Маршруты предназначены для фильтрации поступающих в маршрутизатор событий по различным условиям, что позволяет более гибко выстраивать процесс обработки событий.

На входе маршрутизатора события проверяются по условиям маршрутов и отправляются на выходы тех маршрутов, условиям которых удовлетворяют.

Если событие не удовлетворяет условиям ни одного из маршрутов, то оно направляется на выход маршрутизатора **По умолчанию** (если установлен флажок **Добавить маршрут по умолчанию**) или удаляется (если флажок **Добавить маршрут по умолчанию** не установлен).

В нижней части окна добавления или редактирования маршрутизатора отображается таблица имеющихся в нем маршрутов. Таблица содержит следующие столбцы:

- **Название маршрута** — название маршрута, используемое для его идентификации в системе.
- **Условие** — условие фильтрации исходных событий, заданное на языке VRL.

В VRL-программе для доступа к данным события используется символ точки (**.**), а к метаданным события — символ процента (**%**). В корреляционных событиях метаданные включают данные исходных событий.

Доступные операции над маршрутами:

- Добавление маршрута
- Изменение маршрута
- Удаление маршрута
- Удаление маршрута

Добавление маршрута

Чтобы добавить маршрут:

1. Нажмите на кнопку **Добавить маршрут**. Система отобразит окно добавления маршрута.
2. Введите название маршрута.
3. Введите условие фильтрации исходных событий на языке VRL.

Пример 38. Пример условия

```
.externalId == "4648" && .categoryOutcome == "Success"
```

В редакторе кода доступны горячие клавиши.

4. Нажмите на кнопку **Добавить**. Маршрут будет добавлен в список маршрутов маршрутизатора.

Изменение маршрута

Чтобы изменить маршрут:

1. Нажмите на иконку редактирования в строке требуемого маршрута в списке. Система отобразит окно изменения этого маршрута.
2. Внесите изменения в требуемые поля.
3. Нажмите на кнопку **Сохранить**. Измененные данные маршрута будут сохранены.

Удаление маршрута

Чтобы удалить маршрут, нажмите на иконку удаления в строке требуемого маршрута в списке. Маршрут будет удален из списка маршрутов маршрутизатора без подтверждения действия.

Работа с коррелляторами

Данный раздел описывает процесс работы с коррелляторами на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О коррелляторах

Коррелятор — это элемент конвейера, который представляет собой комплекс правил корреляции. Правила корреляции позволяют обнаруживать и анализировать взаимосвязанные события на основе правил, написанных на языке описания преобразований и корреляций. Эти правила также могут инициировать действия после обнаружения, такие как создание корреляционных событий.

Порядок применения правил корреляции определяется последовательностью их добавления в коррелятор. Таким образом, события проходят через набор правил в установленном порядке, что позволяет осуществлять сложные сценарии корреляции данных.

На диаграмме конфигурации конвейера коррелятор имеет один вход и один выход. Коррелятор на входе применяет правила корреляции к потоку событий. На выходе выдает корреляционные события.

Доступные операции над коррелятором:

- Добавление коррелятора
- Изменение коррелятора
- Удаление коррелятора
- Настройка коррелятора

Режим распределенной корреляции

Императивные правила корреляции не поддерживаются в режиме распределенной корреляции.

Распределенная корреляция использует ресурсы нескольких узлов в кластере для параллельной обработки событий. Данный режим позволяет распределять нагрузку между несколькими экземплярами предпроцессора и коррелятором, увеличивая общую производительность коллектора.

Распределенный коррелятор состоит из двух компонентов: предпроцессора и коррелятора.

Предпроцессор предназначен для первичной обработки событий, поступающих на коррелятор. В предпроцессоре происходит фильтрация событий и создание псевдонимов (aliases). Псевдонимы — это ссылки на события, используемые для упрощения их обработки и анализа. Предпроцессор создает и выпускает корреляционные события. Предпроцессоров может быть несколько, но должен быть минимум один.

Коррелятор в составе распределенного коррелятора отвечает за группировку и составление цепочек событий. Цепочки событий представляют собой последовательности взаимосвязанных событий, обнаруженных на основе логики, определенной в поле `join` правила корреляции. В составе распределенного коррелятора может быть только один коррелятор.

Операции полей `join` и `group` декларативного правила вычисляются на корреляторе. Остальные поля, в том числе `alias` и `filter`, вычисляются на предпроцессорах. События могут испускаться как предпроцессором, так и коррелятором в зависимости от конфигурации правила корреляции.

Для работы коррелятора в распределенном режиме необходимо активировать настройку **Распределенный коррелятор** и выделить дополнительные ресурсы в настройках коллектора. Изменение этой настройки требует перезапуска коллектора.

Добавление коррелятора

Чтобы добавить коррелятор на конфигурацию конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Коррелятор**. Отобразится окно добавления коррелятора.

8. Введите название коррелятора.
9. Задайте интервал очистки событий в секундах. По истечении указанного периода в императивных правилах корреляции будет происходить очистка событий с истекшим временем жизни. Время жизни задается в поле `expire_after_sec` в структуре правила.

   По умолчанию интервал очистки равен 300 секундам.

10. Добавьте в коррелятор правила корреляции.

Если коллектор, на котором размещен коррелятор, настроен для работы в режиме распределенной корреляции, он не поддерживает работу императивных правил корреляции.

   Чтобы добавить правило корреляции:

   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужное правило корреляции.
   b. Установите флажок слева от правила корреляции, которое необходимо добавить в коррелятор.

11. Нажмите на кнопку **Добавить**. Новый коррелятор отобразится на диаграмме конфигурации.

Если в добавленных правилах содержатся отсутствующие активные списки или таблицы обогащения, система отобразит список ошибок с указанием ID и названий правил.

Чтобы добавить конфигурацию с правилами, где обнаружены ошибки, установите флажок **Игнорировать ошибки**. Если флажок не установлен, правила с ошибками будут исключены.

Нажмите на кнопку **Готово**, чтобы сохранить конфигурацию.

Изменение коррелятора

Чтобы изменить коррелятор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Откройте окно настроек коррелятора одним из следующих способов:

- Нажмите на кнопку действий (⋮) справа от названия коррелятора и выберите опцию **Изменить**.
- Дважды нажмите на коррелятор на диаграмме конфигурации конвейера.

8. Внесите изменения в требуемые поля.
9. Нажмите на кнопку **Сохранить**. Измененные данные коррелятора будут сохранены.

Удаление коррелятора

Чтобы удалить коррелятор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку действий (⋮) справа от названия коррелятора и выберите опцию **Удалить**. Система отобразит окно подтверждения удаления.
8. Нажмите на кнопку **Удалить**. Коррелятор будет удален.

Настройка коррелятора

Настройка коррелятора выполняется при его добавлении или изменении.

В окне настроек коррелятора отображается список правил корреляции в выбранном каталоге. Список представлен в виде таблицы со следующими столбцами:

- **ID** — уникальный идентификационный код правила корреляции, присваиваемый системой автоматически при его создании.
- **Название** — имя правила корреляции, используемое для его идентификации в системе.
- **Теги** — пользовательские теги, присвоенные данному правилу корреляции.
- **Версия** — версия правила корреляции, представленная в формате SemVer v2.0.

При работе с таблицей правил корреляции доступны следующие операции:

- Поиск правил по полям **ID**, **Название** и **Теги**.
- Фильтрация правил по тегам. Теги выбираются из выпадающего списка.

  Также доступна фильтрация по наличию правил в корреляторе с помощью флажка **Добавлено в коррелятор**. При установке флажка в таблице отображаются только правила, уже добавленные в коррелятор.

- Сортировка правил по полям **ID**, **Название** и **Версия**.

В окне настроек доступен функционал просмотра и добавления правил корреляции, а также обновления их версии.

Доступные операции по настройке коррелятора:

- Просмотр правила корреляции
- Добавление правил корреляции
- Обновление версии правила корреляции
- Изменение версии правила корреляции

Просмотр правила корреляции

Чтобы просмотреть конфигурацию правила корреляции из окна настройки коррелятора, нажмите на ID соответствующего правила в списке. Система отобразит окно просмотра выбранного правила в новой вкладке браузера.

Добавление правил корреляции

Чтобы добавить правило корреляции в коррелятор:

1. Раскройте дерево каталогов в левой части окна настроек коррелятора и выберите каталог, который содержит нужное правило корреляции.
2. Установите флажок слева от правила корреляции, которое необходимо добавить в коррелятор.

Обновление версии правила корреляции

Чтобы обновить правило корреляции, добавленное в коррелятор, нажмите на кнопку обновления в правой части его строки в списке правил. В рамках коррелятора правило будет обновлено до последней версии.

Кнопка отображается, если существует более новая версия правила, чем добавленная в коррелятор.

Вы можете также выбрать последнюю версию правила в выпадающем списке версий.

Изменение версии правила корреляции

Чтобы установить другую версию правила корреляции:

- Нажмите на кнопку редактирования в правой части строки правила в списке.
- Выберите нужную версию в выпадающем списке и нажмите на кнопку **Сохранить**. В рамках коррелятора будет установлена выбранная версия правила.

Работа с нормализаторами

Данный раздел описывает процесс работы с нормализаторами на конвейерах в системе. Работа

осуществляется при настройке конфигурации конвейера.

О нормализаторах

Нормализатор — это элемент конвейера, который представляет собой комплекс правил нормализации. Каждое такое правило состоит из двух основных компонентов: кода нормализации, выполненного на языке VRL, и фильтра. В случае наличия фильтра, выполнение соответствующего правила нормализации инициируется только при соблюдении условий этого фильтра для входящего события.

Порядок применения правил нормализации определяется последовательностью их добавления в нормализатор. Таким образом, события проходят через набор правил в установленном порядке, что позволяет осуществлять сложные сценарии нормализации данных.

На диаграмме конфигурации конвейера нормализатор имеет один вход и один выход. Нормализатор на входе принимает события, преобразует/обогащает их и выдает преобразованные события на выходе.

Доступные операции над нормализатором:

- Добавление нормализатора
- Изменение нормализатора
- Удаление нормализатора
- Настройка нормализатора

Добавление нормализатора

Чтобы добавить нормализатор на конфигурацию конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Нормализатор**. Отобразится окно добавления нормализатора.
8. Введите название нормализатора.
9. При необходимости из выпадающего списка **Обработка ошибок** выберите шину коллектора, в которую будет отправлено событие в случае возникновения ошибок обработки. По умолчанию выбран вариант **События с ошибкой**.

10. Добавьте в нормализатор правила нормализации.

    Чтобы добавить правило нормализации:

    a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужное правило нормализации.

    b. Установите флажок слева от правила нормализации, которое необходимо добавить в нормализатор.

11. Нажмите на кнопку **Добавить**. Новый нормализатор отобразится на диаграмме конфигурации.

Если в добавленных правилах содержатся отсутствующие активные списки или таблицы обогащения, система отобразит список ошибок с указанием ID и названий правил.

Чтобы добавить конфигурацию с правилами, где обнаружены ошибки, установите флажок **Игнорировать ошибки**. Если флажок не установлен, правила с ошибками будут исключены.

Нажмите на кнопку **Готово**, чтобы сохранить конфигурацию.

Изменение нормализатора

Чтобы изменить нормализатор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Откройте окно настроек нормализатора одним из следующих способов:
   - Нажмите на кнопку действий (⋮) справа от названия нормализатора и выберите опцию **Изменить**.
   - Дважды нажмите на нормализатор на диаграмме конфигурации конвейера.
8. Внесите изменения в требуемые поля.
9. Нажмите на кнопку **Сохранить**. Измененные данные нормализатора будут сохранены.

Удаление нормализатора

Чтобы удалить нормализатор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку действий (⋮) справа от названия нормализатора и выберите опцию **Удалить**. Система отобразит окно подтверждения удаления.
8. Нажмите на кнопку **Удалить**. Нормализатор будет удален.

Настройка нормализатора

Настройка нормализатора выполняется при его добавлении или изменении.

В окне настроек нормализатора отображается список правил нормализации в выбранном каталоге. Список представлен в виде таблицы со следующими столбцами:

- **ID** — уникальный идентификационный код правила нормализации, присваиваемый системой автоматически при его создании.
- **Название** — имя правила нормализации, используемое для его идентификации в системе.
- **Теги** — пользовательские теги, присвоенные данному правилу нормализации.
- **Версия** — версия правила нормализации, представленная в формате SemVer v2.0.

При работе с таблицей правил нормализации доступны следующие операции:

- Поиск правил по полям **ID**, **Название** и **Теги**.
- Фильтрация правил по тегам. Теги выбираются из выпадающего списка.

  Также доступна фильтрация по наличию правил в нормализаторе с помощью флажка **Добавлено в нормализатор**. При установке флажка в таблице отображаются только правила, уже добавленные в нормализатор.

- Сортировка правил по полям **ID**, **Название** и **Версия**.

В окне настроек доступен функционал просмотра и добавления правил нормализации, а также обновления их версии.

Доступные операции по настройке нормализатора:

- Просмотр правила нормализации
- Добавление правил нормализации

- Обновление версии правила нормализации
- Изменение версии правила нормализации

Просмотр правила нормализации

Чтобы просмотреть конфигурацию правила нормализации из окна настройки нормализатора, нажмите на ID соответствующего правила в списке. Система отобразит окно просмотра выбранного правила в новой вкладке браузера.

Добавление правил нормализации

Чтобы добавить правило нормализации в нормализатор:

1. Раскройте дерево каталогов в левой части окна настроек нормализатора и выберите каталог, который содержит нужное правило нормализации.
2. Установите флажок слева от правила нормализации, которое необходимо добавить в нормализатор.

Обновление версии правила нормализации

Чтобы обновить правило нормализации, добавленное в нормализатор, нажмите на кнопку обновления в правой части его строки в списке правил. В рамках нормализатора правило будет обновлено до последней версии.

Кнопка отображается, если существует более новая версия правила, чем добавленная в нормализатор.

Вы можете также выбрать последнюю версию правила в выпадающем списке версий.

Изменение версии правила нормализации

Чтобы установить другую версию правила нормализации:

- Нажмите на кнопку редактирования в правой части строки правила в списке.
- Выберите нужную версию в выпадающем списке и нажмите на кнопку **Сохранить**. В рамках нормализатора будет установлена выбранная версия правила.

Работа с агрегаторами

Данный раздел описывает процесс работы с агрегаторами на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

Об агрегаторах

Агрегатор — это элемент конвейера, который использует правила агрегации для объединения и анализа групп событий безопасности. Каждое правило агрегации включает в себя критерии для группировки событий, условия начала и завершения агрегации, а также стратегии агрегации для каждого поля. Агрегация инициируется в соответствии с условиями, определенными в правилах, и преобразует входящий поток событий в агрегированные данные.

На диаграмме конфигурации конвейера агрегатор имеет один вход и один выход. Агрегатор на входе принимает поток событий, производит их агрегацию и выдает агрегированные события на выходе.

Доступные операции над агрегатором:

- Добавление агрегатора
- Изменение агрегатора
- Удаление агрегатора
- Настройка агрегатора

Добавление агрегатора

Чтобы добавить агрегатор на конфигурацию конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Агрегатор**. Отобразится окно добавления агрегатора.
8. Введите название агрегатора.
9. Добавьте в агрегатор правила агрегации.

   Чтобы добавить правило агрегации:

   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужное правило агрегации.

   b. Установите флажок слева от правила агрегации, которое необходимо добавить в агрегатор.
10. Нажмите на кнопку **Добавить**. Новый агрегатор отобразится на диаграмме конфигурации.

Если в добавленных правилах содержатся отсутствующие активные списки или таблицы обогащения, система отобразит список ошибок с указанием ID и названий правил.

Чтобы добавить конфигурацию с правилами, где обнаружены ошибки, установите флажок **Игнорировать ошибки**. Если флажок не установлен, правила с ошибками будут исключены.

Нажмите на кнопку **Готово**, чтобы сохранить конфигурацию.

Изменение агрегатора

Чтобы изменить агрегатор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Откройте окно настроек агрегатора одним из следующих способов:
   - Нажмите на кнопку действий (⋮) справа от названия агрегатора и выберите опцию **Изменить**.
   - Дважды нажмите на агрегатор на диаграмме конфигурации конвейера.
8. Внесите изменения в требуемые поля.
9. Нажмите на кнопку **Сохранить**. Измененные данные агрегатора будут сохранены.

Удаление агрегатора

Чтобы удалить агрегатор:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку действий (⋮) справа от названия агрегатора и выберите опцию **Удалить**. Система отобразит окно подтверждения удаления.
8. Нажмите на кнопку **Удалить**. Агрегатор будет удален.

Настройка агрегатора

Настройка агрегатора выполняется при его добавлении или изменении.

В окне настроек агрегатора отображается список правил агрегации в выбранном каталоге. Список представлен в виде таблицы со следующими столбцами:

- **ID** — уникальный идентификационный код правила агрегации, присваиваемый системой автоматически при его создании.
- **Название** — имя правила агрегации, используемое для его идентификации в системе.
- **Теги** — пользовательские теги, присвоенные данному правилу агрегации.
- **Версия** — версия правила агрегации, представленная в формате SemVer v2.0.

При работе с таблицей правил агрегации доступны следующие операции:

- Поиск правил по полям **ID**, **Название** и **Теги**.
- Фильтрация правил по тегам. Теги выбираются из выпадающего списка.

  Также доступна фильтрация по наличию правил в агрегаторе с помощью флажка **Добавлено в агрегатор**. При установке флажка в таблице отображаются только правила, уже добавленные в агрегатор.

- Сортировка правил по полям **ID**, **Название** и **Версия**.

В окне настроек доступен функционал просмотра и добавления правил агрегации, а также обновления их версии.

Доступные операции по настройке агрегатора:

- Просмотр правила агрегации
- Добавление правил агрегации
- Обновление версии правила агрегации
- Изменение версии правила агрегации

Просмотр правила агрегации

Чтобы просмотреть конфигурацию правила агрегации из окна настройки агрегатора, нажмите на ID соответствующего правила в списке. Система отобразит окно просмотра выбранного правила в новой вкладке браузера.

Добавление правил агрегации

Чтобы добавить правило агрегации в агрегатор:

1. Раскройте дерево каталогов в левой части окна настроек агрегатора и выберите каталог, который содержит нужное правило агрегации.
2. Установите флажок слева от правила агрегации, которое необходимо добавить в агрегатор.

Обновление версии правила агрегации

Чтобы обновить правило агрегации, добавленное в агрегатор, нажмите на кнопку обновления в правой части его строки в списке правил. В рамках агрегатора правило будет обновлено до последней версии.

Кнопка отображается, если существует более новая версия правила, чем добавленная в агрегатор.

Вы можете также выбрать последнюю версию правила в выпадающем списке версий.

Изменение версии правила агрегации

Чтобы установить другую версию правила агрегации:

- Нажмите на кнопку редактирования в правой части строки правила в списке.
- Выберите нужную версию в выпадающем списке и нажмите на кнопку **Сохранить**. В рамках агрегатора будет установлена выбранная версия правила.

Работа со сбором метрик

Данный раздел описывает процесс работы со сбором метрик на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О сборе метрик

Сбор метрик — это элемент конвейера, который собирает метрики различных элементов конвейеров для их дальнейшей обработки и анализа.

Например, с помощью сбора метрик можно отслеживать поступление событий от источника на конвейер и выявлять ситуации, когда точка входа перестает получать события. Это позволяет своевременно детектировать и исправлять ошибки в работе конвейера.

На диаграмме конфигурации конвейера сбор метрик имеет один выход и не имеет входов. На выходе выдает события, содержащие собранные метрики.

Доступные операции над сбором метрик:

- Добавление сбора метрик
- Изменение сбора метрик
- Удаление сбора метрик

Добавление сбора метрик

Чтобы добавить сбор метрик на конфигурацию конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку ( ) в строке конвейера. Отобразится карточка конвейера.

5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку **Добавить элемент**. Отобразится выпадающее меню элементов конвейера.
8. Выберите пункт **Сбор метрик**. Отобразится окно добавления сбора метрик.
9. Введите название сбора метрик.
10. Задайте интервал между сборами метрик в секундах в поле **Период сбора метрик**. Период можно ввести вручную или задать с помощью кнопок — и +.
11. Выберите элементы для сбора метрик из дерева элементов. Для выбора доступны элементы всех конвейеров в рамках коллектора.

    Чтобы выбрать элемент, установите флажок слева от его названия в списке.

    Элементы в дереве сгруппированы по конвейерам. Чтобы просмотреть список элементов нужного конвейера, нажмите на кнопку ⌄ справа от его названия.

    Чтобы скрыть отображенный список элементов конвейера, нажмите на кнопку ⌃ справа от его названия.
12. Нажмите на кнопку **Добавить**. Новый сбор метрик отобразится на диаграмме конфигурации.

Изменение сбора метрик

Чтобы изменить сбор метрик:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Откройте окно настроек сбора метрик одним из следующих способов:
   - Нажмите на кнопку действий (⋮) справа от названия сбора метрик и выберите опцию **Изменить**.
   - Дважды нажмите на сбор метрик на диаграмме конфигурации конвейера.
8. Внесите изменения в требуемые поля.
9. Нажмите на кнопку **Сохранить**. Измененные данные сбора метрик будут сохранены.

Удаление сбора метрик

Чтобы удалить сбор метрик:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку действий (⋮) справа от названия сбора метрик и выберите опцию **Удалить**. Система отобразит окно подтверждения удаления.
8. Нажмите на кнопку **Удалить**. Сбор метрик будет удален.

Работа с конечными точками

Данный раздел описывает процесс работы с конечными точками на конвейерах в системе. Работа осуществляется при настройке конфигурации конвейера.

О конечных точках

Конечная точка — это элемент конвейера, который отвечает за отправку событий. Тип конечной точки зависит от пункта назначения, в который пересылаются события. На диаграмме конфигурации конвейера конечная точка имеет один вход и не имеет выходов.

В системе доступны следующие типы конечных точек:

- Хранилище событий
- Внешняя система
- Глобальная шина

Доступные операции над конечной точкой:

- Добавление конечной точки
- Изменение конечной точки
- Удаление конечной точки
- Настройка конечной точки

Добавление конечной точки

Чтобы добавить конечную точку на конфигурацию конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Конечная точка**. Отобразится окно добавления конечной точки.
8. При необходимости выберите шаблон для автоматического заполнения полей конечной точки. По умолчанию выбран вариант **Без шаблона**.
9. Заполните поля (набор и содержание полей могут отличаться в зависимости от типа точки):
   - Название конечной точки.
   - Тип конечной точки.
   - Формат конечной точки.
10. Установите флажок **Сохранить как шаблон**, если настройку конечной точки необходимо сохранить в качестве шаблона для дальнейшего использования.
11. Нажмите **Добавить**. Новая конечная точка отобразится на диаграмме конфигурации.

Изменение конечной точки

Чтобы изменить конечную точку:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Откройте окно настроек конечной точки одним из следующих способов:
   - Нажмите на кнопку действий (⋮) справа от названия конечной точки и выберите опцию **Изменить**.

- Дважды нажмите на конечную точку на диаграмме конфигурации конвейера.

8. Внесите изменения в требуемые поля.
9. Нажмите на кнопку **Сохранить**. Измененные данные конечной точки будут сохранены.

Удаление конечной точки

Чтобы удалить конечную точку:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку действий (⋮) справа от названия конечной точки и выберите опцию **Удалить**. Система запросит подтверждение удаления конечной точки.
8. Нажмите на кнопку **Удалить**. Конечная точка будет удалена с конфигурации конвейера.

Настройка конечной точки

Набор параметров конечной точки отличается в зависимости от ее типа. В системе доступны следующие типы конечных точек:

- Хранилище событий
- Внешняя система
- Глобальная шина

Хранилище событий

Конечная точка типа **Хранилище событий** позволяет сохранять события в выбранное хранилище.

Вы можете просматривать события хранилища в разделе **Поиск** или визуализировать их данные с помощью дашбордов.

При создании конечной точки типа **Хранилище событий** требуется задать следующие параметры:

1. **Хранилище событий**: выбор хранилища событий, в которое будут передаваться события.
2. **Режим отладки**: включение режима отладки. В режиме отладки, поля, которые не соответствуют модели события выбранного хранилища, будут принудительно заполнены значениями по умолчанию, при этом в лог будет записано сообщение о наличии проблем в полях исходного события.

Перед тем как переслать события в хранилище, система автоматически производит их нормализацию. С помощью функции `compact` удаляются пустые поля, если они присутствуют в данных:

Пример 39. Код VRL

```
. = compact(.) ?? .
```

Внешняя система

Конечная точка типа **Внешняя система** позволяет отправлять события во внешнюю систему.

При создании конечной точки типа **Внешняя система** требуется выбрать **Тип внешней системы** и задать параметры, зависящие от выбранного типа.

Доступны следующие типы внешних систем:

- Blackhole
- HTTP
- Kafka
- Socket
- Vector
- NATS JetStream

Blackhole

Конечная точка Blackhole предназначена для отбрасывания поступающих событий. Ее рекомендуется использовать для отладки конвейера, когда не нужно сохранять обработанные события в хранилище или отправлять их во внешнюю систему.

Параметры конечной точки Blackhole:

1. **Интервал запроса** (опционально): время в секундах между записями активности конечной точки в логи коллектора.
2. **Количество событий** (опционально): максимальное количество событий в секунду, которые могут обрабатываться конечной точкой.
3. **Использовать расширенные настройки**: настройка буферизации.

HTTP

Конечная точка HTTP предназначена для подключения к внешнему серверу по HTTP-протоколу и отправки событий на этот сервер. Отправка событий происходит за счет выполнения HTTP-запросов, посылаемых от конечной точки к серверу.

Параметры конечной точки HTTP:

1. **Префикс** (опционально): строка, с которой начинается полезная нагрузка (payload) в теле HTTP-запроса.

2. **Суффикс** (опционально): строка, которой заканчивается полезная нагрузка (payload) в теле HTTP-запроса.
3. **Адрес**: адрес внешней системы в формате URL- или IP-адреса с указанием порта. Например: `https://hostname:6000` или `1.0.0.0:6000`.
4. **Тип аутентификации** (опционально): выбор типа аутентификации на HTTP-сервере:
   - **Базовая**: выберите секрет типа **Учетные данные**, хранящий имя пользователя и пароль для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.
   - **Токен**: выберите секрет типа **Токен**, хранящий токен для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.
5. **Тип сжатия**: выбор типа сжатия данных:
   - **NONE** — без сжатия. Данный тип выбран по умолчанию.
   - **GZIP**.
   - **SNAPPY**.
   - **ZLIB**.
   - **ZSTD**.
6. **Формат событий**: формат передачи событий во внешнюю систему. Доступные форматы:
   - **AVRO**. При выборе этого формата требуется ввести схему AVRO в формате JSON. Например:

     ```
     { "type": "record", "name": "log", "fields": [{ "name": "msg", "type": "string"
     }] }
     ```
   - **CSV**. При выборе этого формата требуется ввести поля, которые требуется преобразовать в формат CSV. Например: `dvendor, dproduct, msg`.

     Преобразование в формат CSV не поддерживается для полей с типом **Array** или **KeyValue**. Значения этих полей передаваться не будут.
   - **GELF**.
   - **JSON**. Данный формат выбран по умолчанию.
   - **LOGFMT**.
   - **NATIVE**.
   - **NATIVE_JSON**.
   - **PROTOBUF**. При выборе этого формата требуется загрузить файл с описанием протокола Protocol Buffers (Protobuf). Подробную информацию о структуре файла можно найти в официальной документации Protocol Buffers.
   - **RAW_MESSAGE**.

- **TEXT**.

7. **HTTP-заголовки** (опционально): заголовки HTTP-запроса. Заголовки позволяют уточнить HTTP-запрос и определяют, какие данные требуется передавать в запросе.

   ▼ *Об HTTP-заголовках*

   HTTP-заголовок — это строка в HTTP-запросе, состоящая из пары "имя: значение". Он используется для уточнения запроса или описания содержимого сообщения. Формат заголовка определяется стандартом RFC 822.

   HTTP-заголовки делятся на следующие категории:

   - Основные заголовки (General Headers) — применяются во всех сообщениях клиента и сервера.
   - Заголовки запроса (Request Headers) — используются только в запросах клиента.
   - Заголовки ответа (Response Headers) — используются только в ответах от сервера.
   - Заголовки сущности (Entity Headers) — применяются к содержимому сообщения.

   Полный список HTTP-заголовков можно найти в MDN Web Docs.

   Пример HTTP-заголовка

   Для отправки данных на сервер порциями по 512 байт, в HTTP-запросе необходимо указать заголовок `Content-Length`. Значение этого заголовка должно соответствовать максимальному размеру поступающих данных в байтах, в данном случае это будет `512`.

   Чтобы задать заголовок, нажмите на кнопку **Добавить**. Отобразится строка добавления заголовка. Введите заголовок и его значение. Для одного заголовка можно задать несколько значений.

   Чтобы удалить заголовок, нажмите на кнопку в строке нужного заголовка.

8. **Метод запроса** (опционально): выбор метода отправляемых HTTP-запросов:
   - **DELETE** — удаляет данные с сервера.
   - **GET** — запрашивает данные с сервера с полным HTTP-ответом.
   - **HEAD** — запрашивает данные с сервера без тела HTTP-ответа.
   - **PATCH** — используется для частичного изменения ресурса.
   - **POST** — отправляет данные на сервер.
   - **PUT** — заменяет данные на сервере данными из HTTP-запроса.
   - **OPTIONS** — запрашивает список методов, разрешенных на сервере.
   - **TRACE** — выполняет подключение к серверу и запрашивает от него ответ.
9. **Адрес HTTP-прокси** (опционально): полный адрес для подключения к HTTP-прокси в виде URI. Применяется, если HTTP-сервер использует прокси.
10. **Включить TLS**: включение TLS-сертификата.
11. **Использовать расширенные настройки**: настройка буферизации.

Kafka

1. **Адрес**: IP-адреса серверов с портами для подключения к кластеру Kafka. Адреса вводятся через запятую. Например: `1.0.0.0:9092,1.0.0.1:9092`.
2. **Название топика**: название топика кластера Kafka, в который будут записываться передаваемые события.
3. **Формат событий**: формат передачи событий во внешнюю систему. Доступные форматы:
   - **AVRO**. При выборе этого формата требуется ввести схему AVRO в формате JSON. Например:

     ```
     { "type": "record", "name": "log", "fields": [{ "name": "msg", "type": "string"
     }] }
     ```

   - **CSV**. При выборе этого формата требуется ввести поля, которые требуется преобразовать в формат CSV. Например: `dvendor, dproduct, msg`.

     Преобразование в формат CSV не поддерживается для полей с типом **Array** или **KeyValue**. Значения этих полей передаваться не будут.

   - **GELF**.
   - **JSON**. Данный формат выбран по умолчанию.
   - **LOGFMT**.
   - **NATIVE**.
   - **NATIVE_JSON**.
   - **PROTOBUF**. При выборе этого формата требуется загрузить файл с описанием протокола Protocol Buffers (Protobuf). Подробную информацию о структуре файла можно найти в официальной документации Protocol Buffers.
   - **RAW_MESSAGE**.
   - **TEXT**.
4. **Включить TLS**: включение TLS-сертификата.
5. **Использовать расширенные настройки**: настройка буферизации.

Socket

1. **Адрес**: адрес внешней системы в формате URL- или IP-адреса с указанием порта. Например: `https://hostname:6000` или `1.0.0.0:6000`.
2. **Протокол**: выбор протокола передачи событий. Доступные протоколы:
   - **TCP**. При выборе этого протокола становится доступен флажок **Включить TLS** для настройки TLS-сертификата.
   - **UDP**.
3. **Формат событий**: формат передачи событий во внешнюю систему. Доступные форматы:
   - **AVRO**. При выборе этого формата требуется ввести схему AVRO в формате JSON. Например:

```
{ "type": "record", "name": "log", "fields": [{ "name": "msg", "type": "string"
}] }
```

- **CSV**. При выборе этого формата требуется ввести поля, которые требуется преобразовать в формат CSV. Например: `dvendor, dproduct, msg`.

Преобразование в формат CSV не поддерживается для полей с типом **Array** или **KeyValue**. Значения этих полей передаваться не будут.

- **GELF**.
- **JSON**. Данный формат выбран по умолчанию.
- **LOGFMT**.
- **NATIVE**.
- **NATIVE_JSON**.
- **PROTOBUF**. При выборе этого формата требуется загрузить файл с описанием протокола Protocol Buffers (Protobuf). Подробную информацию о структуре файла можно найти в официальной документации Protocol Buffers.
- **RAW_MESSAGE**.
- **TEXT**.

4. **Включить TLS**: включение TLS-сертификата.
5. **Использовать расширенные настройки**: настройка буферизации.

Vector

1. **Адрес**: адрес внешней системы в формате URL- или IP-адреса с указанием порта. Например: `https://hostname:6000` или `1.0.0.0:6000`.
2. **Включить TLS**: включение TLS-сертификата.
3. **Использовать расширенные настройки**: настройка буферизации.

NATS JetStream

1. **Адрес**: полный путь для подключения к NATS в формате URL- или IP-адреса с указанием порта. Например: `nats://cluster.nats.com` или `nats://127.0.0.1:4242`.

Для обмена сообщениями через NATS по умолчанию используется порт 4222.

2. **Тема**: название темы NATS, в которую будут записываться события.

Тема должна быть создана в NATS до настройки конечной точки.

3. **Формат событий**: формат передачи событий во внешнюю систему. Доступные форматы:
   - **AVRO**. При выборе этого формата требуется ввести схему AVRO в формате JSON. Например:

```
{ "type": "record", "name": "log", "fields": [{ "name": "msg", "type": "string"
}] }
```

- **CSV**. При выборе этого формата требуется ввести поля, которые требуется преобразовать в формат CSV. Например: `dvendor, dproduct, msg`.

  Преобразование в формат CSV не поддерживается для полей с типом **Array** или **KeyValue**. Значения этих полей передаваться не будут.

- **GELF**.
- **JSON**. Данный формат выбран по умолчанию.
- **LOGFMT**.
- **NATIVE**.
- **NATIVE_JSON**.
- **PROTOBUF**. При выборе этого формата требуется загрузить файл с описанием протокола Protocol Buffers (Protobuf). Подробную информацию о структуре файла можно найти в официальной документации Protocol Buffers.
- **RAW_MESSAGE**.
- **TEXT**.

4. **Тип аутентификации** (опционально): выбор типа аутентификации во внешней системе:
   - **Базовая**: выберите секрет типа **Учетные данные**, хранящий имя пользователя и пароль для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   - **Токен**: выберите секрет типа **Токен**, хранящий токен для авторизации.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

   - **NKey**: выберите секрет типа **NATS Key**, хранящий приватный и публичный ключи для подключения к NATS.

     Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Включение TLS-сертификата

Для внешних систем HTTP, Kafka, Socket и Vector становится доступен переключатель **Включить TLS**, позволяющий настроить шифрование TLS.

При включении переключателя необходимо:

1. Выбрать секрет типа **Контейнер PKCS** или **Контейнер PKCS#12**, хранящий данные сертификата TLS.

Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

2. При необходимости выбрать секрет типа **Корневой сертификат**, хранящий данные корневого сертификата.

   Чтобы просмотреть выбранный секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Доступны опции **Подтверждение сертификата** и **Подтверждение имени хоста** для настройки проверки действительности сертификата и соответствия имени удаленного хоста.

Расширенные настройки

Для всех внешних систем доступен переключатель **Использовать расширенные настройки**, позволяющий настроить буферизацию в конечной точке. Буфер используется для накопления событий, когда они не могут быть направлены конечной точкой во внешнюю систему (например, когда события направляются во внешнюю систему быстрее, чем она может их обработать). События находятся в буфере до тех пор, пока их не удастся успешно отправить.

При включении переключателя необходимо:

1. Выбрать тип буфера:
   - **RAM** — для накопления событий используется оперативная память. Тип буфера **RAM** установлен по умолчанию.

     Для этого типа характерны высокая скорость буферизации, но низкая степень сохранности данных в случае сбоев системы. Тип буфера **RAM** рекомендуется использовать, когда требуется обеспечить максимальную производительность буферизации.

     При выборе этого типа требуется указать максимальное количество событий, которое может храниться в буфере.

   - **HDD** — для накопления событий используется дисковое пространство.

     Для этого типа характерны низкая скорость буферизации, но высокая степень сохранности данных за счет их хранения в дисковом пространстве. Тип буфера **HDD** рекомендуется использовать, когда требуется обеспечить сохранность передаваемых данных.

     При выборе этого типа требуется указать размер буфера в байтах. Минимальный размер буфера — приблизительно 256 МБ (268435488 байт).

2. Выбрать поведение буфера при его заполнении:
   - **Ожидание** — поступление событий будет прекращено до освобождения места в буфере. Режим ожидания установлен по умолчанию.

     Режим ожидания гарантирует сохранность данных, но может привести к замедлению работы буферизации.

- **Отбрасывать новые события** — все события, поступающие в буфер после его заполнения, будут отброшены.

Режим отбрасывания новых событий обеспечивает высокую производительность буферизации, но может привести к потере данных.

Глобальная шина

Конечная точка типа **Глобальная шина** позволяет отправлять события в глобальную шину.

Данная конечная точка необходима для работы с коллекторами вне центрального пространства, но также может использоваться и с коллекторами в центральном пространстве.

Необходимость использования глобальной шины для отправки и получения событий из вынесенных коллекторов обусловлена тем, что запись событий в хранилище доступна только на конвейере коллектора, находящегося в центральном пространстве.

При создании конечной точки типа **Глобальная шина** не требуется задавать дополнительных параметров.

Получить события, сохраненные в глобальной шине, можно с помощью точки входа типа Global Bus.

Работа с шинами

Данный раздел описывает процесс работы с шинами коллектора в системе. Работа осуществляется на вкладке **Шины** в карточке коллектора.

О шинах

Шина — это компонент коллектора, который позволяет направлять потоки событий между конвейерами. В коллекторе может быть одна или несколько шин. Шина поддерживает два типа соединения: на получение и на отправку событий. Два конвейера событий могут сообщаться друг с другом посредством шин.

Список шин коллектора отображается на вкладке **Шины** в его карточке. Над списком расположено поле для поиска шин по названию.

Доступные операции над шиной:

- Добавление шины
- Просмотр шины
- Изменение шины
- Удаление шины
- Работа с шиной на конвейере

Добавление шины

Чтобы добавить шину:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Шины** в карточке коллектора. Система отобразит список шин коллектора.
4. В нижней части карточки нажмите на кнопку **Добавить**. На экране отобразится окно добавления шины.
5. Введите название шины.
6. При необходимости введите описание шины.
7. Нажмите на кнопку **Добавить**. Шина будет создана и отобразится в списке шин на вкладке **Шины** в карточке коллектора.

Просмотр шины

Чтобы отобразить информацию о шине:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Шины** в карточке коллектора. Система отобразит список шин коллектора.
4. Нажмите на стрелку (⌄) в строке шины. Отобразится карточка шины.

В верхней части карточки отображается название шины. Справа от названия расположена кнопка ⋮ , по которой открывается выпадающее меню с действиями, доступными для выбранной шины.

Карточка шины содержит следующую информацию:

- **ID** — уникальный идентификационный код шины, генерируемый автоматически при ее создании.
- **Описание** — краткое описание функциональности шины.
- **Дата создания** — дата и время создания шины.
- **Создал** — пользователь, инициировавший создание шины.
- **Дата изменения** — дата и время последнего изменения шины.
- **Изменил** — пользователь, выполнивший последнее изменение шины.

Изменение шины

Чтобы изменить шину:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.

2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Шины** в карточке коллектора. Система отобразит список шин коллектора.
4. Нажмите на кнопку действий (⋮) в строке шины и выберите опцию **Изменить**. Система отобразит окно редактирования шины.
5. Внесите изменения в требуемые поля.
6. Нажмите на кнопку **Сохранить**. Измененные данные шины будут сохранены.

Удаление шины

Чтобы удалить шину:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Шины** в карточке коллектора. Система отобразит список шин коллектора.
4. Нажмите на кнопку действий (⋮) в строке шины и выберите опцию **Удалить**. Система отобразит окно подтверждения действия.
5. Нажмите **Удалить**. Шина будет удалена из коллектора.

Удаление недоступно для базовой шины **События с ошибкой** и для шин, используемых на конвейерах.

Работа с шиной на конвейере

Чтобы отправлять события на шину и/или получать из нее события, требуется добавить соединение с шиной на конфигурацию конвейера.

Существуют следующие типы соединения с шиной на конвейере:

- **Получение** — имеет один вход и не имеет выходов.
- **Отправка** — имеет один выход и не имеет входов.

На конфигурации конвейера шина принимает события на входе соединения типа **Получение** и пересылает их на выход соединения типа **Отправка**. С помощью такой связи посредством использования шины можно настраивать обмен событиями между несколькими конвейерами в рамках одного коллектора.

Доступные операции над шиной на конвейере:

- Добавление шины на конвейер
- Изменение шины на конвейере
- Удаление шины с конвейера

Добавление шины на конвейер

Чтобы добавить соединение с шиной на конфигурацию конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку **Добавить элемент** и выберите из выпадающего списка пункт **Шина**. Отобразится окно добавления соединения с шиной.
8. Выберите из выпадающего списка шину, с которой требуется задать соединение. На выбор доступны только шины текущего коллектора.
9. Введите название соединения с шиной.
10. Выберите из выпадающего списка тип соединения с шиной: **Получение** или **Отправка**.
11. Нажмите на кнопку **Добавить**. Соединение с шиной отобразится на диаграмме конфигурации.

Изменение шины на конвейере

Чтобы изменить настройки шины на конвейере:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Откройте окно настроек соединения с шиной одним из следующих способов:
   - Нажмите на кнопку действий (⋮) справа от названия соединения с шиной и выберите опцию **Изменить**.
   - Дважды нажмите на соединение с шиной на диаграмме конфигурации конвейера.
8. Внесите изменения в требуемые поля.

Тип соединения с шиной (получение/отправка) задается только при ее добавлении на конвейер. При редактировании шины на конвейере, изменение типа ее соединения недоступно.

9. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление.

Удаление шины с конвейера

Чтобы удалить соединение с шиной с конвейера:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Конвейеры** в карточке коллектора. Система отобразит список конвейеров коллектора.
4. Нажмите на стрелку (⌄) в строке конвейера. Отобразится карточка конвейера.
5. Нажмите на кнопку **Конфигурация конвейера** в нижней части карточки. Отобразится диаграмма конфигурации конвейера.
6. Удостоверьтесь, что в выпадающем списке **Версия** на панели инструментов в окне конфигурации конвейера выбран вариант **Черновик**.
7. Нажмите на кнопку действий (⋮) справа от названия соединения с шиной и выберите опцию **Удалить**. Система отобразит окно подтверждения удаления.
8. Нажмите на кнопку **Удалить**. Система удалит соединение с шиной из конфигурации конвейера и отобразит соответствующее уведомление.

Работа с обогащениями

Данный раздел описывает процесс работы с обогащениями коллектора в системе. Работа осуществляется на вкладке **Обогащение** в карточке коллектора.

Об обогащении

Обогащение — это компонент коллектора, который позволяет использовать таблицы обогащения в коррелляторах и/или нормализаторах конвейеров. В коллекторе может быть одно или несколько обогащений. Они используются для обогащения и дополнения данных, передаваемых в полях событий: например, перевода кодов ошибок на понятный пользователю язык или расшифровки аббревиатур в наименовании оборудования.

Список обогащений коллектора отображается на вкладке **Обогащение** в его карточке. Над списком расположено поле для поиска обогащений по названию.

Доступные операции над обогащением:

- Добавление обогащения
- Просмотр обогащения

- Изменение обогащения
- Обновление обогащения
- Удаление обогащения

Добавление обогащения

Чтобы добавить обогащение:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Обогащение** в карточке коллектора. Система отобразит список обогащений коллектора.
4. В нижней части карточки нажмите на кнопку **Добавить**. На экране отобразится окно добавления обогащения.
5. Введите название обогащения.
6. Выберите из выпадающего списка таблицу обогащения. Таблица будет использована для создания обогащения в коллекторе.
7. При необходимости введите описание обогащения.
8. Нажмите на кнопку **Добавить**. Обогащение будет создано и отобразится в списке обогащений на вкладке **Обогащение** в карточке коллектора.

Просмотр обогащения

Чтобы отобразить информацию об обогащении:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Обогащение** в карточке коллектора. Система отобразит список обогащений коллектора.
4. Нажмите на стрелку (⌄) в строке обогащения. Отобразится карточка обогащения.

В верхней части карточки отображается название обогащения. Справа от названия расположена кнопка ⋮ , по которой открывается выпадающее меню с действиями, доступными для выбранного обогащения.

Карточка обогащения содержит следующую информацию:

- **ID обогащения** — уникальный идентификационный код компонента обогащения, генерируемый автоматически при добавлении его в коллектор.
- **Описание** — краткое описание функциональности обогащения.
- **ID таблицы обогащения** — уникальный идентификационный код таблицы обогащения, которая

была использована при создании обогащения.

- **Таблица обогащения** — название таблицы обогащения, которая была использована при создании обогащения.
- **Версия таблицы обогащения** — версия таблицы обогащения, указанная в поле `version` в настройках таблицы.
- **Дата создания** — дата и время создания обогащения.
- **Создал** — пользователь, инициировавший создание обогащения.
- **Дата изменения** — дата и время последнего изменения обогащения.
- **Изменил** — пользователь, выполнивший последнее изменение обогащения.

Изменение обогащения

Чтобы изменить обогащение:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Обогащение** в карточке коллектора. Система отобразит список обогащений коллектора.
4. Нажмите на кнопку действий (⋮) в строке обогащения и выберите опцию **Изменить**. Система отобразит окно редактирования обогащения.
5. Внесите изменения в требуемые поля.

Изменение выбора таблицы обогащения, которая была использована для создания обогащения, недоступно.

6. Нажмите на кнопку **Сохранить**. Измененные данные обогащения будут сохранены.

Обновление обогащения

При обновлении версии таблицы обогащения, становится доступным обновление обогащения, связанного с таблицей, для актуализации его данных.

Чтобы обновить обогащение:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Обогащение** в карточке коллектора. Система отобразит список обогащений коллектора.
4. Нажмите на кнопку **Обновить** в строке нужного обогащения. Кнопка отображается только в том случае, если версия таблицы обогащения, связанной с обогащением, обновлена.

Данные обогащения будут актуализированы в соответствии с новой версией таблицы обогащения.

Удаление обогащения

Чтобы удалить обогащение:

1. Перейдите в раздел **Ресурсы** → **Коллекторы**. Система отобразит сведения об имеющихся коллекторах.
2. Нажмите на строку коллектора в списке. Система отобразит в правой части экрана карточку этого коллектора с подробной информацией о нем.
3. Перейдите на вкладку **Обогащение** в карточке коллектора. Система отобразит список обогащений коллектора.
4. Нажмите на кнопку действий (⋮) в строке обогащения и выберите опцию **Удалить**. Система отобразит окно подтверждения действия.
5. Нажмите **Удалить**. Обогащение будет удалено из коллектора.

Удаление доступно только для обогащений, таблицы обогащения которых не задействованы в правилах корреляции или нормализации, используемых в корреляторах или нормализаторах на конвейерах текущего коллектора.

Работа с моделями событий

Данный раздел описывает процесс работы с моделями событий в системе. Работа осуществляется в разделе **Ресурсы** → **Модели событий** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О моделях событий

Модель события определяет формат хранилища событий — имена, типы полей и типы данных в полях. При создании хранилища событий обязательно указывается используемая в нем модель события.

Система поддерживает модели событий двух типов:

- **Системные модели событий** — создаются системой автоматически. Они недоступны для изменения или удаления.
- **Пользовательские модели событий** — создаются вручную.

В системе имеются следующие системные модели событий:

- **Универсальная модель события** — предназначена для хранения информации о событиях в унифицированном виде.
- **Модель события аудита** — предназначена для хранения информации о событиях сервиса аудита.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новую модель события.
- Поле **Поиск** предназначено для быстрого поиска моделей событий в отображаемом списке по названию. Выпадающий список **Статус** позволяет настроить отображение списка моделей событий в соответствии с выбранным статусом.
- Кнопки **Экспорт** (⤒) и **Импорт** (⤓) позволяют выгружать существующие модели событий из системы и загружать в систему модели событий извне.

В рабочей области отображается таблица имеющихся моделей событий. При выборе конкретной модели в рабочей области отображается таблица ее полей, которая включает в себя следующие столбцы:

- **Ключ поля** — уникальный ключ идентификации поля модели события.
- **Описание** — краткое описание функциональности поля модели события.
- **Тип поля** — тип поля:
  - **Служебное** — поле, создаваемое системой автоматически.
  - **Регулярное** — поле, добавленное в модель события вручную.
- **Тип данных** — тип данных значений, хранящихся в поле модели события.
- **Параметры типа данных** — дополнительные параметры значений поля, зависящие от выбранного типа данных.
- **Поле хранящее название** — ключ поля, определяющего семантику названия текущего поля.
- **Может быть null** — возможность хранить значение типа `null` в поле модели события.
- **Отображать поле** — отображение поля при просмотре списка событий модели.

Над таблицей расположены инструменты для работы с моделью события:

- Переключатель статуса модели.
- Кнопка для редактирования модели.
- Кнопка удаления модели события.
- Кнопка создания новой модели события на базе текущей.
- Кнопка для экспорта модели события.
- Поле поиска полей модели по их ключу.

В рабочей области раздела отображается таблица имеющихся моделей событий. Таблица содержит следующие столбцы:

- **Название** — название модели события, используемое для ее идентификации в системе.
- **Описание** — описание модели события.
- **Тип** — тип модели события: системная или пользовательская.
- **Статус** — текущий статус модели события (включена или выключена).

- **Дата создания** — дата и время создания модели события.
- **Дата изменения** — дата и время последнего изменения модели события.

При работе с таблицей моделей событий доступны следующие операции:

- Поиск моделей по полю **Название**.
- Фильтрация моделей по полю **Статус**. Статус выбирается из выпадающего списка.
- Сортировка моделей по полям **Название**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

Работа с моделью события

Доступные операции над моделями событий:

- Создание модели события
- Просмотр информации о модели события
- Включение модели события
- Изменение модели события
- Импорт модели события
- Экспорт модели события
- Создание новой модели события на базе текущей
- Удаление модели события

Создание модели события

Чтобы создать модель события:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания модели события.
3. Введите название модели события.
4. Введите описание модели события.
5. В нижней части окна находится раздел **Схема**, содержащий список полей, которые будут использоваться в этой модели события.

   По умолчанию система отображает набор **служебных** полей.

   Список можно расширить, добавив вручную **регулярные** поля. Для этого:

   a. Нажмите на кнопку **Добавить поле**. Система отобразит окно добавления поля.

   b. Введите уникальный ключ идентификации нового поля.

Ключ может состоять только из латинских букв, цифр и подчеркиваний. При этом он не может начинаться с цифры.

c. При необходимости установите флажок **Поле с изменяемой семантикой**, чтобы использовать значения из другого поля при построении семантики названия нового поля.

Выберите поле из выпадающего списка **Поле, хранящее название**, чтобы определить, какое поле будет использовать система для формирования его названия. Можно использовать только следующие типы полей: **String**, **LCString**, **Enum**.

Пример

Создадим поле с ключом `collector_number` и укажем, что для задания названия этого поля нужно использовать поле `id_list`. Предположим, поле `id_list` принимает значение `collector_id`, а поле `collector_number` — `250`.

При применении этой модели события для поля с ключом `collector_number` в событии отобразится значение: `collector_id:250`.

d. Введите описание нового поля.

e. Выберите тип данных нового поля. В зависимости от выбранного типа данных необходимо указать его параметры:

- **Array** — выберите тип элемента **Array** из списка.
- **Enum** — укажите значения Enum, разделяя их запятой без пробелов.
- **KeyValue** — выберите тип элемента **KeyValue** из списка.
- **UInt** — выберите разрядность из списка.

f. При необходимости включите переключатель **Может быть null**, чтобы поле могло хранить значение типа `null`. Если переключатель выключен, а поле не заполнено, событие в хранилище не будет создано.

В полях типов **Array**, **KeyValue**, **LCString** этот переключатель не используется.

g. При необходимости включите переключатель **Отображать поле**, чтобы поле отображалось в списке результатов поиска в разделе **Поиск**.

Эту настройку можно изменить при просмотре модели события или при настройке внешнего вида списка событий.

h. Нажмите на кнопку **Добавить**.

Повторите действия, чтобы добавить несколько регулярных полей.

По умолчанию, универсальная модель содержит 150 служебных полей. Пользователь может добавить до 20 регулярных полей.

При необходимости вы можете изменить или удалить добавленные регулярные поля:

- Чтобы изменить поле, нажмите на кнопку в строке поля. Отобразится окно изменения поля, в котором внесите требуемые изменения и нажмите на кнопку **Сохранить**. Измененные данные поля будут сохранены.

- Чтобы удалить поле, нажмите на кнопку в строке поля. Поле будет удалено из списка полей модели события.

> Служебные поля в модели данных создаются и заполняются автоматически и недоступны для изменения или удаления.

6. Нажмите на кнопку **Создать**. Система создаст модель событий и отобразит соответствующее уведомление. Новая модель появится в списке раздела **Ресурсы** → **Модели событий**.

Просмотр информации о модели события

Чтобы просмотреть информацию о модели события:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на строку модели события в списке. Система отобразит в рабочей области список полей, содержащихся в этой модели.

Название модели события отображается в верхней части рабочей области.

Чтобы закрыть список полей модели и вернуться в раздел **Ресурсы** → **Модели событий**, нажмите на кнопку **Назад**, расположенную над списком, или перейдите в раздел **Ресурсы** → **Модели событий** на боковой панели.

Список полей модели содержит следующую информацию о каждом поле:

- Ключ поля.
- Описание поля.
- Тип поля: служебное (создается системой автоматически) или регулярное (создается вручную).
- Тип данных значений поля и его параметры.
- Поле, определяющее семантику названия текущего поля.
- Возможность хранить значение `null` в поле.
- Отображение поля при просмотре события модели. Чтобы поле отображалось, переведите переключатель в активное положение. По умолчанию отображаются следующие поля:
  - для универсальной модели события: `timestamp`, `dvendor`, `dproduct`, `name`, `msg`, `dvc`, `outcome`;
  - для модели события аудита: `timestamp`, `raw`, `suser`, `eventType`, `moduleKey`, `eventRiskLevel`, `data`, `meta`;
  - для пользовательской модели события: `id`, `type`, `timestamp`, `sourceIp`.

Чтобы найти поле модели, введите его ключ в строку поиска над списком полей модели и нажмите клавишу ENTER. В списке отобразятся поля модели, в ключе которых встречается введенный запрос.

Включение модели события

> Включение и выключение доступно только для моделей событий с типом **Пользовательская**.

Для того чтобы модель события можно было использовать при создании хранилища событий, ее нужно включить.

При создании модель события включается автоматически.

Чтобы включить модель события:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на строку модели события в списке. Система отобразит в рабочей области список полей, содержащихся в этой модели.
3. Переведите переключатель состояния модели события, расположенный над списком полей модели, в активное положение. Система включит модель и отобразит соответствующее уведомление. Модель события появится в списке доступных моделей при создании хранилища событий.

Чтобы выключить работающую модель события:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на строку модели события в списке. Система отобразит в рабочей области список полей, содержащихся в этой модели.
3. Переведите переключатель состояния модели события, расположенный над списком полей модели, в неактивное положение. Отобразится окно подтверждения выключения модели события.
4. Нажмите на кнопку **Подтвердить**. Система выключит модель и отобразит соответствующее уведомление. Модель события исчезнет из списка доступных моделей при создании хранилища событий.

Изменение модели события

Изменение доступно только для моделей событий с типом **Пользовательская**.

Чтобы изменить модель события:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на строку модели события в списке. Система отобразит в рабочей области список полей, содержащихся в этой модели.
3. Нажмите на кнопку , расположенную над списком полей модели. Отобразится окно редактирования модели события.
4. Внесите требуемые изменения в настройки модели события.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке и списке моделей.

Импорт модели события

Для загрузки модели события в систему применяется функционал импорта, поддерживающий обработку файлов в формате JSON.

Чтобы импортировать модель событий:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на кнопку ⤓ на панели инструментов. Система отобразит окно импорта модели события.
3. Нажмите **Выбрать файл** (⊕) и выберите файл модели в формате JSON.
4. Нажмите на кнопку **Импортировать**. Начнется валидация и импорт моделей. Система отобразит окно с итогами импорта.

Модель будет успешно импортирована, если соответствует следующим критериям:

- Идентификатор модели не совпадает с идентификаторами моделей, уже существующих в системе.
- Ключи полей уникальны и содержат только латинские буквы, цифры и подчеркивания, при этом не начинаются с цифры.
- В полях указаны корректные значения.
- Поля типов **LCString**, **Array** и **KeyValue** не содержат пустых значений.
- Поля с измененной семантикой ссылаются на существующие поля.

Экспорт модели события

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на строку модели события в списке. Система отобразит в рабочей области список полей, содержащихся в этой модели.
3. Нажмите на кнопку ⤒, расположенную над списком полей модели. Система загрузит данные модели на устройство пользователя в формате JSON и отобразит соответствующее уведомление.

Создание новой модели события на базе текущей

Чтобы создать новую модель события на базе текущей:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на строку модели события в списке. Система отобразит в рабочей области список полей, содержащихся в этой модели.
3. Нажмите на кнопку ＋, расположенную над списком полей модели. Отобразится окно создания новой модели события, в котором автоматически будут добавлены все поля текущей модели.
4. Введите название модели события.

5. Введите описание модели события.
6. При необходимости измените состав полей новой модели события.
7. Нажмите на кнопку **Создать**. Система создаст модель события и отобразит соответствующее уведомление. Новая модель появится в списке раздела **Ресурсы** → **Модели событий**.

Удаление модели события

Удаление доступно только для моделей событий с типом **Пользовательская**.

Удаление недоступно для моделей событий, используемых в других сущностях системы.

Чтобы удалить модель события:

1. Перейдите в раздел **Ресурсы** → **Модели событий**. Система отобразит сведения об имеющихся моделях событий.
2. Нажмите на строку модели события в списке. Система отобразит в рабочей области список полей, содержащихся в этой модели.
3. Нажмите на кнопку , расположенную над списком полей модели. Отобразится окно подтверждения удаления модели события.
4. Нажмите на кнопку **Удалить**. Система удалит модель события и отобразит соответствующее уведомление. Удаленная модель будет исключена из списка.

Универсальная модель события

Универсальная модель события предназначена для хранения информации о событиях в едином, универсальном, виде. Универсальная модель позволяет приводить поступающие в систему базовые события к унифицированному формату с единым набором полей.

Разложение данных базовых событий по полям универсальной модели происходит в ходе процесса нормализации, выполняющегося при работе конвейера обработки событий.

Универсальная модель события является встроенной моделью и недоступна для изменения или удаления. Она отображается в разделе **Ресурсы** → **Модели событий**.

При необходимости вы можете создать собственную (пользовательскую) модель события с требуемым набором полей.

Никакое поле универсальной модели события не может хранить в себе дополнительное значение типа `null`. Чтобы хранить такое значение, требуется создать собственную модель события и добавить в нее поле с установленным переключателем **Может быть null**.

Поля универсальной модели разделяются на следующие группы:

- Служебные поля
- Регулярные поля
- Регулярные настраиваемые поля

- Поля активов

Служебные поля

Служебные поля — обязательные поля модели события, создаваемые системой автоматически. Служебные поля образуют минимальный набор полей, необходимый для получения, хранения, анализа, обработки и отправки событий.

Служебные поля по умолчанию содержатся во всех моделях событий и не могут быть изменены или удалены.

Таблица 6. Служебные поля универсальной модели события

Ключ поля	Тип данных	Заполняется системой по умолчанию	Описание
`id`	UUID	Да	Уникальный идентификатор события. Генерируется коллектором после нормализации.
`sourceIp`	IPv6	Нет	IP-адрес источника, от которого получено базовое событие.
`tenantId`	LCString	Да	Идентификатор тенанта.
`art`	DateTime	Нет	Дата и время получения события системой-источником или агентом.
`raw`	String	Нет	Текст базового (сырого) события.
`collectorId`	LCString	Да	Идентификатор коллектора, получившего событие.
`timestamp`	DateTime	Да	Дата и время получения события коллектором.
`aggregationRuleName`	LCString	Нет	Название правила агрегации, обработавшего событие.
`aggregationRuleId`	LCString	Нет	Идентификатор правила агрегации, обработавшего событие.
`type`	Enum	Нет	Тип события. Возможные значения: • -1 — не указано; • 0 — базовое событие; • 1 — нормализованное событие; • 2 — агрегированное событие; • 3 — корреляционное событие; • 4 — событие аудита.

Ключ поля	Тип данных	Заполняется системой по умолчанию	Описание
`cnt`	UInt32	Нет	Количество базовых событий, на основе которых создано агрегированное событие.
`correlationRuleName`	LCString	Нет	Имя правила корреляции, на основе которого создано корреляционное событие.
`correlationRuleId`	LCString	Нет	Идентификатор правила корреляции, на основе которого создано корреляционное событие.
`baseEventIds`	Массив UUID	Нет	Список идентификаторов базовых событий, на основе которых создано агрегированное или корреляционное событие.
`baseEventTimestamps`	Массив DateTime	Нет	Список дат и времен получения базовых событий в коллекторе, на основе которых создано агрегированное или корреляционное событие. Используется для идентификации событий, связанных с корреляционным или агрегированным событием.
`groupedBy`	Массив LCString	Нет	Список полей модели события, по которому группируются события.
`correlationSeverity`	Enum	Нет	Уровень угрозы корреляционного события. Возможные значения: • -1 — не указано; • 1 — низкий; • 2 — средний; • 3 — высокий; • 4 — критический.

Регулярные поля

Регулярные поля — дополнительные поля модели события, не являющиеся обязательными. В пользовательских моделях событий регулярными полями являются все поля, добавленные пользователем при создании или редактировании модели.

В отличие от служебных полей, набор регулярных полей не является универсальным и может изменяться от модели к модели.

Таблица 7. Регулярные поля универсальной модели события

Ключ поля	Тип данных	Описание
`aid`	LCString	Идентификатор агента, от которого получено событие.

Ключ поля	Тип данных	Описание
`at`	LCString	Тип агента, от которого получено событие. Например: "Endpoint".
`av`	LCString	Версия агента, от которого получено событие.
`dvendor`	LCString	Название производителя источника журнала. Значение берется из базового события.
`dproduct`	LCString	Название продукта из источника журнала. Значение берется из базового события.
`dversion`	LCString	Версия продукта из источника журнала Значение берется из базового события.
`deviceEventClassId`	LCString	Идентификатор типа события из источника журнала. Некоторые источники журнала определяют категорию событий.
`name`	String	Название события. Значение берется из базового события.
`severity`	LCString	Уровень важности события. Значение берется из базового события.
`act`	LCString	Действие, которое предпринято устройством. Например: `blocked`, `detected`.
`app`	LCString	Протокол уровня приложений. Например: HTTP, HTTPS, Telnet.
`cat`	LCString	Категория базового события. Устройства часто используют собственную схему категоризации для классификации событий.
`destinationDnsDomain`	LCString	DNS-часть полного доменного имени (FQDN) точки назначения, если базовое событие содержит сведения об отправителе и получателе данных.
`destinationServiceName`	LCString	Название сервиса на принимающей стороне. Например: "sshd". Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.

Ключ поля	Тип данных	Описание
destinationTranslatedAddress	IPv6	IP-адрес устройства, принимающего трафик (после трансляции). Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
destinationTranslatedPort	UInt16	Номер порта на устройстве приемника трафика (после трансляции адреса приемника). Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
deviceDirection	Enum	Направление соединения из базового события. Значения: • -1 — не указано; • 0 — входящее соединение; • 1 — исходящее соединение.
deviceDnsDomain	LCString	DNS-часть полного доменного имени (FQDN) IP-адреса устройства, с которого пришло базовое событие.
deviceExternalId	LCString	Внешний идентификатор устройства. Значение при его наличии берется из базового события.
deviceFacility	LCString	Значение параметра Facility. Значение при его наличии берется из базового события. Например: в Syslog в параметре Facility может передаваться название компоненты ОС, в которой произошла ошибка.
deviceInboundInterface	LCString	Название интерфейса входящего соединения.
deviceNtDomain	LCString	Доменное имя Windows-устройства.
deviceOutboundInterface	LCString	Название интерфейса исходящего соединения.
devicePayloadId	String	Идентификатор полезной нагрузки, который ассоциирован с базовым событием.
deviceProcessName	LCString	Название процесса. Значение берется из базового события.
deviceTranslatedAddress	IPv6	Ретранслированный IP-адрес устройства, с которого пришло базовое событие.

Ключ поля	Тип данных	Описание
`dhost`	LCString	Название хоста, принимающего трафик. Полное доменное имя (FQDN) приемника трафика, если доступно. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`dmac`	MAC	MAC-адрес устройства, принимающего трафик. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`dntdom`	LCString	Доменное имя Windows-устройства, принимающего трафик. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`dpid`	UInt64	Идентификатор системного процесса, ассоциированного с приемником трафика в базовом событии. Например: если в событии указано `Process ID 105`, то `destinationProcessId=105`. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`dpriv`	String	Названия security-ролей, которые определяют пользовательские привилегии на стороне точки назначения. Например: "User", "Guest", "Administrator". Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`dproc`	LCString	Название системного процесса в точке назначения. Например: "sshd", "telnet". Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`dpt`	Uint16	Номер порта на стороне точки назначения. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
`dst`	IPv6	IPv6-адрес точки назначения. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.

Ключ поля	Тип данных	Описание
dtz	LCString	Часовой пояс устройства, на котором сгенерировано событие.
duid	LCString	Идентификатор пользователя на стороне точки назначения. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
duser	LCString	Имя пользователя на стороне точки назначения. Может содержать адрес электронной почты пользователя. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
dvc	IPv6	IPv6-адрес устройства, с которого получено событие.
dvchost	LCString	Название хоста устройства, с которого получено событие. Полное доменное имя (FQDN) устройства, если доступно.
dvcmac	MAC	MAC-адрес устройства, с которого получено событие.
dvcpid	UInt64	Идентификатор системного процесса устройства, сгенерировавшего событие.
endTime	DateTime	Дата и время завершения действия. Значение берется из базового события.
externalId	LCString	Идентификатор устройства, сгенерировавшего событие.
fileCreateTime	DateTime	Дата и время создания файла. Значение берется из базового события.
fileHash	String	Хеш-код файла.
fileId	LCString	Идентификатор файла.
fileModificationTime	DateTime	Дата и время последнего изменения файла.
filePath	String	Путь к файлу (с указанием имени файла).
filePermission	String	Список разрешений к файлу.
fileType	LCString	Тип файла. Например: application, pipe, socket.
fname	String	Имя файла (без указания пути к файлу).
fsize	UInt64	Размер файла.
bytesIn	UInt64	Количество байтов, полученных из источника и переданных получателю. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.

Ключ поля	Тип данных	Описание
msg	String	Краткое описание ошибки (проблемы) из события.
oldFileCreateTime	DateTime	Дата и время создания OLD-файла из события.
oldFileHash	String	Хеш-код OLD-файла.
oldFileId	LCString	Идентификатор OLD-файла (при наличии).
oldFileModificationTime	DateTime	Дата и время последнего изменения OLD-файла.
oldFileName	String	Имя OLD-файла (без указания пути к файлу).
oldFilePath	String	Путь к OLD-файлу (с указанием имени файла).
oldFilePermission	String	Список разрешений к OLD-файлу.
oldFileSize	UInt64	Размер OLD-файла.
oldFileType	LCString	Тип OLD-файла. Например: application, pipe, socket.
bytesOut	UInt64	Количество отправленных байтов. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
outcome	LCString	Результат выполнения действия act на устройстве.
proto	LCString	Название протокола 4-уровня OSI. Например: TCP, UDP.
reason	String	Причина появления события. Например: "bad password" или "unknown user". Также может содержать ошибку или код. Например: “0x1234”
request	String	Запрошенный URL-адрес.
requestClientApplication	String	Агент, который обработал запрос request.
requestContext	String	Описание контекста запроса.
requestCookies	String	Файлы cookie, связанные с запросом.
requestMethod	LCString	Метод, который использовался для доступа к веб-адресу. Например: POST, GET.
rt	DateTime	Дата и время получения устройством сведений о наблюдаемом событии.

Ключ поля	Тип данных	Описание
shost	LCString	Название хоста источника трафика. Полное доменное имя (FQDN) источника трафика, если доступно. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
smac	MAC	MAC-адрес устройства источника трафика. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
sntdom	LCString	Доменное имя Windows-устройства источника трафика. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
sourceDnsDomain	LCString	DNS-часть полного доменного имени (FQDN) Windows-устройства источника трафика. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
sourceServiceName	LCString	Название сервиса на стороне источника трафика. Например: "sshd". Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
sourceTranslatedAddress	IPv6	IPv6-адрес перехода источника. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
sourceTranslatedPort	UInt16	Номер порта перехода на стороне источника. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
spid	UInt64	Идентификатор системного процесса, ассоциированного с источником трафика в базовом событии. Например: если в событии указано `Process ID 105`, то `SourceProcessId=105`. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.

Ключ поля	Тип данных	Описание
spriv	String	Названия security-ролей, которые определяют пользовательские привилегии на стороне источника. Например: "User", "Guest", "Administrator". Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
sproc	String	Название системного процесса на стороне источника. Например: "sshd", "telnet". Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
spt	UInt16	Номер порта на стороне источника. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
src	IPv6	IPv6-адрес источника. Используется для обработки логов сетевого трафика, где необходимо отличать источник и точку назначения.
startTime	DateTime	Дата и время начала действия.
suid	LCString	Идентификатор пользователя на стороне источника.
suser	LCString	Имя пользователя на стороне источника. Может содержать адрес электронной почты пользователя.
status	LCString	Статус события или действия из источника.
logonType	String	Тип входа.
sourceSessionId	String	Идентификатор сессии источника события.
targetSessionId	String	Идентификатор целевой сессии события.
cmd	String	Команда в командной строке.
logonProcessName	LCString	Имя процесса авторизации.
authPackageName	LCString	Имя пакета авторизации.
objType	LCString	Тип объекта действия.
objName	String	Имя объекта действия.
objPath	String	Путь к объекту действия.
accessMask	LCString	Запрошенная маска доступа.
accessList	LCString	Список запрошенных прав доступа.
accesses	LCString	Запрошенные права доступа в читаемом формате.
dGroup	LCString	Имя целевой группы SID (объекта) действия.
dGroupDom	LCString	Имя домена целевой группы SID (объекта) действия.

Регулярные настраиваемые поля

Регулярные настраиваемые поля — это особый тип регулярных полей в универсальной модели события, который не привязан к какому-то конкретному способу применения. Эти поля позволяют хранить в модели события данные того или иного типа, не предусмотренные заранее прочими полями модели.

Пример:

В событии передается адрес электронной почты пользователя, совершившего подозрительное действие. По умолчанию в универсальной модели события для хранения электронной почты нет специальных полей. В таком случае можно воспользоваться регулярными настраиваемыми полями. Данные об адресе электронной почты можно записать в поле `cs1`, а в поле `cs1Label` указать предназначение поля `cs1`, например "E-mail пользователя".

Таблица 8. Регулярные настраиваемые поля универсальной модели события

Ключ поля	Тип данных	Описание
`cn1`	UInt64	Поле для маппинга целочисленного значения, которое не может быть сопоставлено любому другому полю модели события.
`cn1Label`	LCString	Поле, хранящее название для `cn1`.
`cn2`	UInt64	Поле для маппинга целочисленного значения, которое не может быть сопоставлено любому другому полю модели события.
`cn2Label`	LCString	Поле, хранящее название для `cn2`.
`cn3`	UInt64	Поле для маппинга целочисленного значения, которое не может быть сопоставлено любому другому полю модели события.
`cn3Label`	LCString	Поле, хранящее название для `cn3`.
`cn4`	UInt64	Поле для маппинга целочисленного значения, которое не может быть сопоставлено любому другому полю модели события.
`cn4Label`	LCString	Поле, хранящее название для `cn4`.
`cn5`	UInt64	Поле для маппинга целочисленного значения, которое не может быть сопоставлено любому другому полю модели события.
`cn5Label`	LCString	Поле, хранящее название для `cn5`.
`cs1`	String	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
`cs1Label`	LCString	Поле, хранящее название для `cs1`.
`cs2`	String	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.

Ключ поля	Тип данных	Описание
cs2Label	LCString	Поле, хранящее название для cs2.
cs3	String	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
cs3Label	LCString	Поле, хранящее название для cs3.
cs4	String	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
cs4Label	LCString	Поле, хранящее название для cs4.
cs5	String	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
cs5Label	LCString	Поле, хранящее название для cs5.
cs6	String	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
cs6Label	LCString	Поле, хранящее название для cs6.
cs7	String	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
cs7Label	LCString	Поле, хранящее название для cs7.
cs8	String	Поле для маппинга строкового значения, которое не может быть сопоставлено любому другому полю модели события.
cs8Label	LCString	Поле, хранящее название для cs8.
deviceCustomDate1	DateTime	Поле для маппинга значения даты и времени, которое не может быть сопоставлено любому другому полю модели события.
deviceCustomDate1Label	LCString	Поле, хранящее название для deviceCustomDate1.
deviceCustomDate2	DateTime	Поле для маппинга значения даты и времени, которое не может быть сопоставлено любому другому полю модели события.
deviceCustomDate2Label	LCString	Поле, хранящее название для deviceCustomDate2.
deviceCustomDate3	DateTime	Поле для маппинга значения даты и времени, которое не может быть сопоставлено любому другому полю модели события.
deviceCustomDate3Label	LCString	Поле, хранящее название для deviceCustomDate3.
c6a1	IPv6	Поле для маппинга значения IPv6 (или IPv4 в нотации IPv6), которое не может быть сопоставлено любому другому полю модели события.

Ключ поля	Тип данных	Описание
c6a1Label	LCString	Поле, хранящее название для c6a1.
c6a2	IPv6	Поле для маппинга значения IPv6 (или IPv4 в нотации IPv6), которое не может быть сопоставлено любому другому полю модели события.
c6a2Label	LCString	Поле, хранящее название для c6a2.
c6a3	IPv6	Поле для маппинга значения IPv6 (или IPv4 в нотации IPv6), которое не может быть сопоставлено любому другому полю модели события.
c6a3Label	LCString	Поле, хранящее название для c6a3.
c6a4	IPv6	Поле для маппинга значения IPv6 (или IPv4 в нотации IPv6), которое не может быть сопоставлено любому другому полю модели события.
c6a4Label	LCString	Поле, хранящее название для c6a4.

Поля активов

Поля активов — это особый тип регулярных полей, позволяющий хранить сведения об активах организации. Активы представляют собой ресурсы организации, используемые в ее IT-инфраструктуре: например, оборудование, персонал.

Таблица 9. Поля активов универсальной модели события

Ключ поля	Тип данных	Описание
sourcePersonID	LCString	Уникальный идентификатор пользователя на стороне источника.
sourcePersonFullName	LCString	ФИО пользователя на стороне источника.
sourcePersonPosition	LCString	Должность пользователя на стороне источника.
sourcePersonUserName	LCString	Учетная запись пользователя на стороне источника. Может содержать несколько значений.
sourceDeviceID	LCString	Уникальный идентификатор оборудования на стороне источника.
sourceDeviceName	LCString	Название оборудования на стороне источника.
sourceDeviceIPAddress	IPv6	IP-адрес оборудования на стороне источника.
sourceDeviceOS	LCString	Операционная система оборудования на стороне источника.
sourceDeviceOwner	LCString	Владелец оборудования на стороне источника.
targetPersonID	LCString	Уникальный идентификатор пользователя на стороне точки назначения.
targetPersonFullName	LCString	ФИО пользователя на стороне точки назначения.
targetPersonPosition	LCString	Должность пользователя на стороне точки назначения.

Ключ поля	Тип данных	Описание
`targetPersonUserName`	LCString	Учетная запись пользователя на стороне точки назначения. Может содержать несколько значений.
`targetDeviceID`	LCString	Уникальный идентификатор оборудования на стороне точки назначения.
`targetDeviceName`	LCString	Название оборудования на стороне точки назначения.
`targetDeviceIPAddress`	IPv6	IP-адрес оборудования на стороне точки назначения.
`targetDeviceOS`	LCString	Операционная система оборудования на стороне точки назначения.
`targetDeviceOwner`	LCString	Владелец оборудования на стороне точки назначения.

Типы данных

Раздел содержит описания типов данных, поддерживаемых в системе, с особенностями их использования и примерами.

Таблица 10. Поддерживаемые типы данных

Тип данных	Описание	Пример
`DateTime`	Дата и время. Допустимые форматы значения: • `DD.MM.YYYY HH:MM:SS`, где: ◦ `DD.MM.YYYY` — дата в формате "день.месяц.год". ◦ `HH:MM:SS` — время в формате "час:минута:секунда". • `YYYY-MM-DDThh:mm:ss+/-hh:mm`, где: ◦ `YYYY-MM-DD` — дата в формате "год-месяц-день". ◦ `T` — начало обозначения времени. ◦ `hh:mm:ss` — время в формате "час:минута:секунда". ◦ `+hh:mm` или `-hh:mm` — разница между местным временем и часовым поясом UTC+0. При использовании формата `DD.MM.YYYY HH:MM:SS` время указывается относительно часового пояса пользователя. Формат отображения дат и времени в системе задается в настройках профиля. В полях событий дата и время могут задаваться следующими способами: • `"<DateTime-значение>"` — используется для хранения даты-времени в виде строки. Например: `"15.05.2024 06:28:15"`. • `t'<DateTime-значение>'` — используется для возможности корреляции данных по дате-времени. Например: `t'15.05.2024 06:28:15'`.	15.05.2024 06:28:15 2024-05-15T10:28:15-04:00
`String`	Строка. Задается в двойных кавычках.	"абвг123"

Тип данных	Описание	Пример
`LCString`	Строка со словарным кодированием. Задается в двойных кавычках. Строковое значение записывается в словарь и заменяется на индекс его записи в словаре. При небольших размерах словаря (например, менее 10 000 записей) данные типа `LCString` обрабатываются быстрее по сравнению с типом `String`.	"абв"
`UInt`	Неотрицательное целое число фиксированной длины. Для определения диапазона значений требуется указать разрядность. Диапазоны значений, определяемые разрядностью: • `UInt8`: [0 : 2^{8}-1]; • `UInt16`: [0 : 2^{16}-1]; • `UInt32`: [0 : 2^{32}-1]; • `UInt64`: [0 : 2^{64}-1]; • `UInt256`: [0 : 2^{256}-1].	255
`Float`	Число с плавающей точкой. Для определения диапазона значений требуется указать разрядность. Приблизительные диапазоны значений, определяемые разрядностью: • `Float32`: [-3,4x10^{38} : 3,4x10^{38}]; • `Float64`: [-1,7x10^{308} : 1,7x10^{308}].	0.999
`UUID`	Универсальный уникальный идентификатор (UUID). Значение — пять групп из шестнадцатеричных цифр в формате `8-4-4-4-12`, разделенных дефисом. Буквы — только строчные латинские. Формат значения: `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`	61f0c404-5cb3-11e7-907b-a6006ad3dba0
`Bool`	Логическое значение.	True, False

Тип данных	Описание	Пример
`KeyValue`	Пара `ключ:значение`. Элементы пары указываются в фигурных скобках и разделяются двоеточием. Для ключа допускается только тип `String`. Для значения доступны следующие типы: • `DateTime`; • `String`; • `LCString`; • `UInt`; • `Float32`; • `UUID`; • `IPv6`; • `IPv4`; • `MAC`.	{"key1":"1"}
`Enum`	Список предопределенных значений. Может хранить до 256 значений, пронумерованных в диапазоне [-128 : 127]. Элементы списка указываются в круглых скобках и разделяются запятой и пробелом.	('high_severity', 'medium_severity', 'low_severity')
`IPv4`	IPv4-адрес. Значение — четыре группы из трех десятичных цифр, разделенных точкой. Формат значения: `xxx.xxx.xxx.xxx`	10.99.101.99

Тип данных	Описание	Пример
`IPv6`	IPv6-адрес. Значение по стандарту RFC 5952 — восемь групп из четырех шестнадцатеричных цифр, разделенных двоеточием. Буквы — только строчные латинские. Формат значения: `xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx` Допускается использование сжатой формы записи IPv6 в соответствии со следующими правилами: 1. В каждой группе ведущие нули (нули слева) удаляются. 2. Если группа состоит из одних нулей, она заменяется на один ноль. 3. Одна непрерывная последовательность из нулей (самая длинная) заменяется символом двойного двоеточия (`::`). Может применяться только один раз. Допускается хранение IPv4 в нотации IPv6: 1. Последние 32 младших бита записываются как IPv4-адрес. 2. В начале идут только нулевые поля и одно единичное, записанные в нотации IPv6.	2a01:1f34::8a2e:765d 2a02:e980:1e::1
`MAC`	MAC-адрес. Значение — шесть двузначных шестнадцатеричных чисел, разделенных двоеточием. Буквы — только заглавные латинские. Формат значения: `XX:XX:XX:XX:XX:XX`	00:53:00:B8:DF:B8

Тип данных	Описание	Пример
`Array`	Массив значений заданного типа данных. Элементы массива указываются в квадратных скобках и разделяются запятой и пробелом. Допустимые типы данных значений массива: • `DateTime`; • `String`; • `LCString`; • `UInt`; • `Float32`; • `UUID`; • `IPv6`; • `IPv4`; • `MAC`.	[321, 123, 12344]

Работа с базами данных событий

Данный раздел описывает процесс работы с базами данных событий в системе. Работа осуществляется в разделе **Ресурсы** → **Базы данных событий** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О базах данных событий

Базы данных (БД) событий используются для хранения событий в системе. Единицей хранения событий в БД является хранилище событий: на уровне БД хранилище представляет собой таблицу. Столбцы и типы данных хранилища задаются моделью события.

Система поддерживает БД событий двух типов:

- **Системные БД событий** — регистрируются системой автоматически. Они недоступны для изменения или удаления.

Системные БД не могут быть использованы для создания хранилищ событий и активных списков.

- **Пользовательские БД событий** — регистрируются вручную.

В системе имеется системная БД событий аудита. Эта БД лежит в основе системного хранилища событий, предназначенного для хранения событий сервиса аудита.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Зарегистрировать** (+) позволяет зарегистрировать новую БД событий.
- Поле **Поиск** предназначено для быстрого поиска БД событий в отображаемом списке по названию.

В рабочей области отображается таблица имеющихся БД событий. При выборе конкретной БД в правой части рабочей области отображается ее карточка с детальной информацией, которая включает в себя следующие вкладки:

1. **Информация** — отображает информацию о БД событий. Содержит следующие поля:
   - **CPU** — количество ядер и уровень загруженности процессора сервиса.
   - **RAM** — объем и уровень загруженности оперативной памяти сервиса.
   - **HDD** — объем и уровень загруженности жесткого диска сервиса.
   - **ID** — уникальный идентификационный код БД, генерируемый автоматически при ее создании.
   - **Название** — имя БД, используемое для ее идентификации в системе.
   - **Идентификатор сервиса** — уникальный идентификационный код БД в системе обнаружения сервисов NATS Service Discovery.
   - **ID тенанта** — уникальный идентификационный код тенанта, к которому относится БД.
   - **Тип** — тип БД событий: системная или пользовательская.
   - **Описание** — краткое описание функциональности БД.
   - **Статус** — текущий статус БД (активна или недоступна).
   - **Хранилищ событий в БД** — количество хранилищ событий, содержащихся в БД.
   - **Дата создания** — дата и время создания БД.
   - **Создал** — пользователь, инициировавший создание БД.
   - **Дата изменения** — дата и время последнего изменения БД.
   - **Изменил** — пользователь, выполнивший последнее изменение БД.
   - **EPS за последние 30 минут** — график с количеством зарегистрированных событий в секунду (EPS) за последние 30 минут с шагом в 5 минут.
2. **Тома** — отображает информацию о томах в БД событий. Доступные тома: hot_volume (горячее хранение) — для быстрого доступа к часто используемым данным, cold_volume (холодное хранение) — для хранения редко используемых данных.

   Для каждого тома отображается следующая информация:

   - **Свободно/всего** — информация о свободном и общем объеме тома.
   - **Уведомлять, если свободно менее** — порог свободного объема тома, при достижении которого система сформирует предупреждения в карточке БД и отобразит иконки ⓘ в таблице раздела.

     Для настройки порога свободного объема в томе предназначена кнопка **Настройка тома**.

При исчерпании всего свободного объема тома происходит остановка конвейеров, связанных с текущей БД событий.

В нижней части вкладки расположена кнопка для настройки хранения данных в БД по умолчанию.

В верхнем правом углу карточки расположена кнопка действий (⋮). При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранной БД событий: изменение и удаление.

В рабочей области раздела отображается таблица имеющихся БД событий. Таблица содержит следующие столбцы:

- **Название** — название БД, используемое для ее идентификации в системе.
- **Статус** — текущий статус БД (активна или недоступна).
- **Идентификатор сервиса** — уникальный идентификатор БД в системе обнаружения сервисов NATS Service Discovery.
- **Описание** — описание БД.
- **Дата создания** — дата и время создания БД.
- **Дата изменения** — дата и время последнего изменения БД.

Если в томах базы данных заканчивается свободное пространство, слева от ее записи в таблице отображается иконка ⓘ.

При работе с таблицей БД событий доступны следующие операции:

- Поиск баз данных по полю **Название**.
- Сортировка баз данных по полям **Название**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

Работа с базами данных событий

Доступные операции над БД событий:

- Регистрация базы данных событий
- Просмотр базы данных событий
- Изменение базы данных событий
- Удаление базы данных событий

Регистрация базы данных событий

Чтобы зарегистрировать БД событий:

1. Перейдите в раздел **Ресурсы** → **Базы данных событий**. Система отобразит сведения об имеющихся БД событий.
2. Нажмите на кнопку **Зарегистрировать** (+). Отобразится окно регистрации БД событий.

3. Введите название БД событий.
4. Введите идентификатор БД ClickHouse в NATS Service Discovery.

Чтобы получить идентификатор БД ClickHouse в NATS Service Discovery, обратитесь к администратору системы или воспользуйтесь разделом Получение идентификатора сервиса в NATS Service Discovery.

5. При необходимости введите описание БД событий.
6. Нажмите на кнопку **Зарегистрировать**. Система создаст БД событий и отобразит соответствующее уведомление. Новая база отобразится в списке раздела **Ресурсы → Базы данных событий**.

Просмотр базы данных событий

Чтобы просмотреть БД событий:

1. Перейдите в раздел **Ресурсы → Базы данных событий**. Система отобразит сведения об имеющихся БД событий.
2. Нажмите на строку БД событий в списке. Система отобразит в правой части экрана карточку этой БД с подробной информацией о ней.

В верхней части карточки отображается название БД событий.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

По умолчанию при выборе БД событий открывается вкладка **Информация**. На вкладке отображается информация о выбранной БД.

Вкладка содержит следующие параметры:

- Характеристика процессора (CPU), объем оперативной памяти (RAM) и жесткого диска (HDD) сервиса, а также их уровень загруженности.
- Идентификатор БД.
- Название БД.
- Идентификатор БД в системе обнаружения сервисов NATS Service Discovery.
- Идентификатор тенанта, к которому относится БД.
- Тип БД: системная или пользовательская.
- Описание БД.
- Статус БД.
- Количество хранилищ, содержащихся в БД.
- Дата и время создания БД.
- Идентификатор пользователя, создавшего БД.
- Дата и время последнего изменения БД.
- Идентификатор пользователя, изменившего БД.

- График с количеством зарегистрированных событий в секунду (EPS) за последние 30 мин. с шагом в 5 мин.

Изменение базы данных событий

Изменение доступно только для БД событий с типом **Пользовательская**.

Чтобы изменить БД событий:

1. Перейдите в раздел **Ресурсы** → **Базы данных событий**. Система отобразит сведения об имеющихся БД событий.
2. Нажмите на строку БД событий в списке. Система отобразит в правой части экрана карточку этой БД с подробной информацией о ней.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно изменения настроек БД.
4. Внесите изменения в требуемые поля БД событий.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке и списке БД событий.

Изменить название и описание БД событий можно непосредственно из ее карточки:

1. Перейдите в раздел **Ресурсы** → **Базы данных событий**. Система отобразит сведения об имеющихся БД событий.
2. Нажмите на строку БД событий в списке. Система отобразит в правой части экрана карточку этой БД с подробной информацией о ней.
3. Убедитесь, что в карточке БД событий открыта вкладка **Информация**.
4. Нажмите на кнопку рядом с текущим названием или описанием БД событий в ее карточке. Система отобразит поле ввода значения.
5. Введите новое значение поля.
6. Нажмите **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке и списке БД событий.

Удаление базы данных событий

Удаление доступно только для БД событий с типом **Пользовательская**.

Удаление недоступно для БД, используемых в других сущностях системы.

Чтобы удалить БД:

1. Перейдите в раздел **Ресурсы** → **Базы данных событий**. Система отобразит сведения об имеющихся БД событий.
2. Нажмите на строку БД событий в списке. Система отобразит в правой части экрана карточку этой БД с подробной информацией о ней.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления БД.

4. Нажмите на кнопку **Удалить**. Система удалит БД событий и отобразит соответствующее уведомление. Удаленная база будет исключена из списка.

Тома хранения данных

Тома хранения данных в БД событий бывают следующих типов:

- **hot_volume** — предназначен для быстрого доступа к часто используемым данным (горячее хранение).
- **cold_volume** — предназначен для хранения редко используемых данных (холодное хранение).

Доступные операции над томами БД событий:

- Просмотр томов базы данных событий
- Настройки томов базы данных событий
- Настройка хранения данных в базе данных событий по умолчанию

Просмотр томов базы данных событий

Чтобы отобразить информацию о томах БД событий:

1. Перейдите в раздел **Ресурсы** → **Базы данных событий**. Система отобразит сведения об имеющихся БД событий.
2. Нажмите на строку БД событий в списке. Система отобразит в правой части экрана карточку этой БД с подробной информацией о ней.
3. Перейдите на вкладку **Тома** в карточке БД событий. Система отобразит список томов БД.

Вкладка содержит следующую информацию о томах hot_volume (горячее хранение) и cold_volume (холодное хранение):

- **Свободно/всего** — информация о свободном и общем объеме тома.
- **Уведомлять, если свободно менее** — порог свободного объема тома, при достижении которого система сформирует предупреждения в карточке БД и отобразит иконки ⓘ в таблице раздела.

  Для настройки порога свободного объема в томе предназначена кнопка **Настройка тома**.

При исчерпании всего свободного объема тома происходит остановка конвейеров, связанных с текущей БД событий.

В нижней части вкладки расположена кнопка для настройки хранения данных в БД по умолчанию.

Настройки томов базы данных событий

Чтобы настроить том базы данных событий:

1. Перейдите в раздел **Ресурсы** → **Базы данных событий**. Система отобразит сведения об имеющихся БД событий.
2. Нажмите на строку БД событий в списке. Система отобразит в правой части экрана карточку этой БД с подробной информацией о ней.

3. Перейдите на вкладку **Тома** в карточке БД событий. Система отобразит список томов БД.
4. Нажмите на кнопку **Настроить том** требуемого тома. Отобразится окно настройки тома.
5. Задайте пороговое значение свободного объема тома, при достижении которого система сформирует уведомление. Доступно задание относительного (в процентах) и абсолютного (в ГБ) свободного объема.
6. Нажмите на кнопку **Сохранить**. Система сохранит настройки тома базы данных и отобразит соответствующее уведомление.

Настройка хранения данных в базе данных событий по умолчанию

Настройки хранения данных в БД событий по умолчанию можно использовать для быстрой настройке хранения данных в хранилищах событий.

Чтобы настроить хранение данные в БД событий по умолчанию:

1. Перейдите в раздел **Ресурсы → Базы данных событий**. Система отобразит сведения об имеющихся БД событий.
2. Нажмите на строку БД событий в списке. Система отобразит в правой части экрана карточку этой БД с подробной информацией о ней.
3. Перейдите на вкладку **Тома** в карточке БД событий. Система отобразит список томов БД.
4. Нажмите на кнопку **Настроить хранение по умолчанию** в нижней части карточки БД событий. Отобразится окно настроек хранения данных по умолчанию.
5. Задайте настройки хранения данных:
   a. Выберите из выпадающего списка том хранения данных:
      - **hot_volume** — том для быстрого доступа к часто используемым данным (горячее хранение).
      - **cold_volume** — том для хранения редко используемых данных (холодное хранение).
   b. Укажите срок хранения данных в днях.

      Если поле для срока хранения данных оставить пустым, то срок хранения данных в этой БД будет неограниченным.

   c. Выберите из выпадающего списка действие над данными в томе по окончании срока хранения:
      - **Переместить** — данные будут перемещены в другой том. Система добавит дополнительную строку для настройки нового тома, в который требуется переместить данные.
      - **Удалить** — данные в томе будут удалены.
6. Нажмите на кнопку **Сохранить**. Система сохранит настройки хранения данных в БД по умолчанию и отобразит соответствующее уведомление.

Работа с хранилищами событий

Данный раздел описывает процесс работы с хранилищами событий в системе. Работа

осуществляется в разделе **Ресурсы** → **Хранилища событий** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О хранилищах событий

Хранилище событий используется для хранения событий, поступивших в систему и обработанных коллектором. Оно представляет собой таблицу в БД событий. Формат хранилища определяется его моделью данных.

Система поддерживает хранилища событий двух типов:

- **Системные хранилища событий** — создаются системой автоматически. Они недоступны для изменения названия/описания или удаления. Системные хранилища не могут быть использованы в качестве хранилищ событий для конечных точек.
- **Пользовательские хранилища событий** — создаются вручную.

В системе имеется системное хранилище событий аудита, предназначенное для хранения событий сервиса аудита. Это хранилище построено на системной БД событий аудита.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новое хранилище событий.
- Поле **Поиск** предназначено для быстрого поиска хранилищ событий в отображаемом списке по названию.

В рабочей области отображается таблица имеющихся хранилищ событий. При выборе конкретного хранилища в правой части рабочей области отображается его карточка с детальной информацией, которая включает в себя следующие вкладки:

1. **Информация** — отображает информацию о хранилищах событий. Содержит следующие поля:
    - **ID** — уникальный идентификационный код хранилища событий, генерируемый автоматически при его создании.
    - **Название** — имя хранилища событий, используемое для его идентификации в системе.
    - **Описание** — краткое описание функциональности хранилища событий.
    - **ID тенанта** — уникальный идентификационный код тенанта, к которому относится хранилище.
    - **Тип** — тип хранилища событий: системное или пользовательское.
    - **БД событий** — БД событий, в которой содержится хранилище.
    - **Модель событий** — модель события, определяющая формат хранилища.
    - **Событий в хранилище** — количество событий в хранилище.
    - **Дата создания** — дата и время создания хранилища событий.

- **Создал** — пользователь, инициировавший создание хранилища событий.
- **Дата изменения** — дата и время последнего изменения хранилища событий.
- **Изменил** — пользователь, выполнивший последнее изменение хранилища событий.

На вкладке также отображаются показатели сжатия данных, хранящихся в таблице ClickHouse, которая ассоциирована с хранилищем событий:

- объем занимаемого пространства без сжатия;
- объем занимаемого пространства со сжатием;
- коэффициент сжатия, показывающий отношение занимаемого объема без сжатия к занимаемому объему со сжатием.

Для получения более подробной информации о сжатии предназначена кнопка **Подробнее**.

2. **Тома** — отображает информацию о томах в хранилище событий. Доступные тома: hot_volume (горячее хранение) — для быстрого доступа к часто используемым данным, cold_volume (холодное хранение) — для хранения редко используемых данных.

   Для каждого тома отображается следующая информация:

   - **Свободно/всего** — информация о свободном и общем объеме тома.
   - **Количество событий в томе** — информация о количестве событий, хранящихся в томе.
   - **Объем занимаемого места в томе** — информация о фактически занимаемом объеме тома.
   - **Уведомлять, если свободно менее** — порог свободного объема тома, при достижении которого система уведомляет пользователя с помощью предупреждения в карточке хранилища и иконки ⓘ в таблице раздела.

   Настройка уведомления при достижении порога свободного объема тома выполняется в карточке БД событий, лежащей в основе текущего хранилища.

   При исчерпании всего свободного объема тома происходит остановка конвейеров, связанных с текущим хранилищем событий.

   - **Срок хранения** — количество дней, в течение которых события хранятся в томе.

В верхнем правом углу карточки расположена кнопка действий (⋮). При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранного хранилища событий: изменение и удаление.

В рабочей области раздела отображается таблица имеющихся хранилищ событий. Таблица содержит следующие столбцы:

- **Название** — название хранилища событий, используемое для его идентификации в системе.
- **Описание** — описание хранилища событий.
- **БД событий** — БД событий, в которой содержится хранилище.
- **Модель событий** — модель события, используемая хранилищем.

- **Событий в хранилище** — количество событий в хранилище.
- **Дата создания** — дата и время создания хранилища событий.
- **Дата изменения** — дата и время последнего изменения хранилища событий.

Если в томах хранилища заканчивается свободное место, слева от его записи в таблице отображается иконка .

При работе с таблицей хранилищ событий доступны следующие операции:

- Поиск хранилищ по полю **Название**.
- Сортировка хранилищ по полям **Название**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

Работа с хранилищем событий

Доступные операции над хранилищами событий:

- Создание хранилища событий
- Просмотр хранилища событий
- Просмотр показателей сжатия по полям
- Изменение хранилища событий
- Удаление хранилища событий

Создание хранилища событий

Чтобы создать хранилище событий:

1. Перейдите в раздел **Ресурсы** → **Хранилища событий**. Система отобразит сведения об имеющихся хранилищах событий.
2. Нажмите на кнопку **Создать** (+). Отобразится окно создания хранилища событий.
3. Выберите из выпадающего списка БД событий, в которую требуется поместить хранилище. Система отобразит в нижней части поля для настройки томов хранения этой БД.

В системных базах данных создание хранилищ событий не поддерживается.

4. Выберите из выпадающего списка модель события, которая будет использована для создаваемого хранилища.
5. Введите название хранилища событий.
6. При необходимости введите описание хранилища событий.
7. Задайте настройки хранения данных:

   a. Выберите том хранения данных:

      - **hot_volume** — том для быстрого доступа к часто используемым данным (горячее хранение).

- **cold_volume** — том для хранения редко используемых данных (холодное хранение).

b. Укажите срок хранения данных в днях.

Если поле для срока хранения данных оставить пустым, то срок хранения данных в этом хранилище будет неограниченным.

c. Выберите из выпадающего списка действие над данными в томе по окончании срока хранения:

- **Переместить** — данные будут перемещены в другой том. Введите в появившейся строке настройки нового тома, в который требуется переместить данные.
- **Удалить** — данные в томе будут удалены.

При необходимости можно задать настройки хранения данных на основе настроек по умолчанию в выбранной БД. Для этого нажмите на кнопку **Применить настройки хранения по умолчанию БД**. Если настройки по умолчанию не заданы, система отобразит соответствующее сообщение.

8. Нажмите на кнопку **Создать**. Система создаст хранилище событий и отобразит соответствующее уведомление. Новое хранилище появится в списке раздела **Ресурсы → Хранилища событий**.

Просмотр хранилища событий

Чтобы просмотреть хранилище событий:

1. Перейдите в раздел **Ресурсы → Хранилища событий**. Система отобразит сведения об имеющихся хранилищах событий.
2. Нажмите на строку хранилища событий в списке. Система отобразит в правой части экрана карточку этого хранилища с подробной информацией о нем.

В верхней части карточки отображается название хранилища событий.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

По умолчанию при выборе хранилища событий открывается вкладка **Информация**. На вкладке отображается информация о выбранном хранилище. Вкладка содержит следующие параметры:

- Идентификатор хранилища.
- Название хранилища.
- Описание хранилища.
- Идентификатор тенанта, к которому относится хранилище.
- Тип хранилища: системное или пользовательское.
- БД событий, в которой создано хранилище.
- Модель данных, используемая для хранилища.
- Количество событий в хранилище.
- Дата и время создания хранилища.
- Идентификатор пользователя, создавшего хранилище.

- Дата и время последнего изменения хранилища.
- Идентификатор пользователя, изменившего хранилище.

На вкладке также отображаются показатели сжатия данных, хранящихся в таблице ClickHouse, которая ассоциирована с хранилищем событий:

- объем занимаемого пространства без сжатия;
- объем занимаемого пространства со сжатием;
- коэффициент сжатия, показывающий отношение занимаемого объема без сжатия к занимаемому объему со сжатием.

Для получения более подробной информации о сжатии предназначена кнопка **Подробнее**.

Просмотр показателей сжатия по полям

Основная информация о сжатии данных в хранилище отображается на вкладке **Информация** в карточке хранилища.

Чтобы получить подробную информацию о показателях сжатия по каждому полю в модели события, лежащей в основе хранилища:

1. Перейдите в раздел **Ресурсы** → **Хранилища событий**. Система отобразит сведения об имеющихся хранилищах событий.
2. Нажмите на строку хранилища событий в списке. Система отобразит в правой части экрана карточку этого хранилища с подробной информацией о нем.
3. Убедитесь, что в карточке хранилища открыта вкладка **Информация**.
4. Нажмите на кнопку **Подробнее** справа от заголовка группы **Показатели сжатия** в карточке. Откроется окно показателей сжатия по полям.

В верхней части окна отображаются основные показатели сжатия данных, которые хранятся в таблице ClickHouse, ассоциированной с хранилищем событий:

- объем занимаемого пространства без сжатия;
- объем занимаемого пространства со сжатием;
- коэффициент сжатия, показывающий отношение занимаемого объема без сжатия к занимаемому объему со сжатием.

Также в окне представлена таблица показателей сжатия по всем полям модели события. Таблица содержит следующие столбцы:

- **Название** — название поля в модели события.
- **Объем без сжатия** — объем занимаемого полем пространства без сжатия.
- **Объем со сжатием** — объем занимаемого полем пространства со сжатием.
- **Коэффициент сжатия** — отношение занимаемого полем объема без сжатия к занимаемому объему со сжатием.

- Показатели сжатия по полям отображаются только после фактического

выполнения оптимизации в хранилище и расчета показателей сжатия для каждого столбца.

- Из-за особенностей оптимизации столбцов в ClickHouse в ряде случаев показатели сжатия по полям могут быть равны 0.

При работе с таблицей доступны следующие операции:

- Поиск показателей по полю **Название**.
- Сортировка показателей по любому столбцу.

Вы также можете экспортировать таблицу показателей сжатия в файл формата CSV по кнопке ↑.

Изменение хранилища событий

Изменение названия и описания доступно только для хранилищ событий с типом **Пользовательское**.

Чтобы изменить хранилище событий:

1. Перейдите в раздел **Ресурсы** → **Хранилища событий**. Система отобразит сведения об имеющихся хранилищах событий.
2. Нажмите на строку хранилища событий в списке. Система отобразит в правой части экрана карточку этого хранилища с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно изменения настроек хранилища.
4. Внесите изменения в требуемые поля хранилища событий.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке и списке хранилищ.

Изменить название и описание хранилища можно непосредственно из его карточки:

1. Перейдите в раздел **Ресурсы** → **Хранилища событий**. Система отобразит сведения об имеющихся хранилищах событий.
2. Нажмите на строку хранилища событий в списке. Система отобразит в правой части экрана карточку этого хранилища с подробной информацией о нем.
3. Убедитесь, что в карточке хранилища событий открыта вкладка **Информация**.
4. Нажмите на кнопку рядом с текущим названием или описанием хранилища в его карточке. Система отобразит поле ввода значения.
5. Введите новое значение поля.
6. Нажмите **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке и списке хранилищ.

Удаление хранилища событий

Удаление доступно только для хранилищ событий с типом **Пользовательское**.

Удаление недоступно для хранилищ событий, используемых в других сущностях системы.

Чтобы удалить хранилище событий:

1. Перейдите в раздел **Ресурсы** → **Хранилища событий**. Система отобразит сведения об имеющихся хранилищах событий.
2. Нажмите на строку хранилища событий в списке. Система отобразит в правой части экрана карточку этого хранилища с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления хранилища.
4. Нажмите на кнопку **Удалить**. Система удалит хранилище событий и отобразит соответствующее уведомление. Удаленное хранилище будет исключено из списка.

Тома хранения данных

Тома хранения данных подгружаются в хранилище событий автоматически из БД событий, в которой содержится хранилище.

Тома хранения данных в хранилище событий бывают следующих типов:

- **hot_volume** — предназначен для быстрого доступа к часто используемым данным (горячее хранение).
- **cold_volume** — предназначен для хранения редко используемых данных (холодное хранение).

Просмотр томов хранилища событий

Чтобы отобразить информацию о томах хранилища событий:

1. Перейдите в раздел **Ресурсы** → **Хранилища событий**. Система отобразит сведения об имеющихся хранилищах событий.
2. Нажмите на строку хранилища событий в списке. Система отобразит в правой части экрана карточку этого хранилища с подробной информацией о нем.
3. Перейдите на вкладку **Тома** в карточке хранилища событий. Система отобразит список томов хранилища.

Вкладка содержит следующую информацию о томах hot_volume (горячее хранение) и cold_volume (холодное хранение):

- **Свободно/всего** — информация о свободном и общем объеме тома.
- **Количество событий в томе** — информация о количестве событий, хранящихся в томе.
- **Объем занимаемого места в томе** — информация о фактически занимаемом объеме тома.
- **Уведомлять, если свободно менее** — порог свободного объема тома, при достижении которого система уведомляет пользователя с помощью предупреждения в карточке хранилища и иконки ⓘ в таблице раздела.

Настройка уведомления при достижении порога свободного объема тома выполняется в карточке БД событий, лежащей в основе текущего хранилища.

При исчерпании всего свободного объема тома происходит остановка конвейеров, связанных с текущим хранилищем событий.

- **Срок хранения** — количество дней, в течение которых события хранятся в томе.

Работа с активными списками

Данный раздел описывает процесс работы с активными списками в системе. Работа осуществляется в разделе **Ресурсы** → **Активные списки** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Об активных списках

Активный список представляет собой двумерный массив данных для накопления информации, полученной из определенных полей событий, поступающих на коррелятор.

Активные списки предназначены для повышения эффективности корреляционных правил, упрощения процедур обогащения событий и сбора информации для последующего анализа. Они используются в системе для обеспечения перекрестной ссылки в процессе корреляционного анализа, выполняемого корреляторами.

Один и тот же список может быть использован в разных корреляторах системы. При этом данные в активный список можно добавлять только через элементы экспертизы в корреляторе, на котором применяется список.

В каждом списке пользователь системы имеет возможность задать конкретный состав полей и параметры хранения данных с помощью схемы активного списка. Эти списки могут содержать отдельные поля событий и обработанные данные, полученные из этих полей.

Для управления активными списками предусмотрены функции для получения, добавления и удаления записей. В ходе работы системы записи могут быть добавлены или удалены из активных списков с помощью этих функций.

Данные, занесенные в активный список, доступны для чтения и изменения другим элементам экспертизы, которые используют язык VRL.

Активные списки в системе используются в двух ключевых сценариях:

- Реализация многоуровневых корреляционных сценариев, где одно правило обновляет список, а другое осуществляет его анализ.
- Использование в качестве справочных таблиц: загрузка эталонных данных и их вызов в рамках определенного правила.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (＋) позволяет создать новый активный список.
- Поле **Поиск** предназначено для быстрого поиска активных списков в отображаемом списке по названию.

В рабочей области отображается таблица имеющихся активных списков. При выборе конкретного активного списка на экране открывается карточка с детальной информацией о нем.

Таблица содержит следующие столбцы:

- **ID** — уникальный идентификационный код активного списка, генерируемый автоматически при его создании.
- **Схема** — название схемы, которая определяет конфигурацию и параметры активного списка.
- **База данных** — название базы данных событий, используемой активным списком.

  В системных базах данных активные списки создавать нельзя.

- **TTL** — время жизни записей в активном списке в секундах. Если имеет значение "0", то время жизни записей неограниченно.
- **Лимит записей** — максимально допустимое количество записей в активном списке.
- **Дата создания** — дата и время создания активного списка.
- **Дата изменения** — дата и время последнего изменения активного списка.
- **Ресурсы** — количество ресурсов системы, использующих активный список. При наведении курсора на количество ресурсов отображается всплывающее сообщение с полным перечнем связанных с активным списком ресурсов.
- **Версия схемы** — версия схемы активного списка, представленная в формате SemVer v2.0.

При работе с таблицей активных списков доступны следующие операции:

- Поиск активных списков по полю **Схема**.
- Сортировка активных списков по полям **Схема**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

Работа с активными списками

Доступные операции над активными списками:

- Создание активных списков
- Просмотр активных списков
- Изменение активных списков
- Удаление активных списков

Создание активных списков

Чтобы создать активный список:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся

активных списках.

2. Нажмите на кнопку **Создать** ( ). Отобразится окно создания активного списка.
3. Выберите из выпадающего списка схему активного списка. В окне отобразятся сведения о выбранной схеме:
   - время жизни записей в секундах (TTL);
   - лимит записей;
   - поля активного списка, определенные схемой.
4. Выберите из выпадающего списка базу данных событий.
5. При необходимости введите описание активного списка.
6. Нажмите на кнопку **Создать**. Система создаст активный список и отобразит соответствующее уведомление. Новый активный список появится в разделе **Ресурсы** → **Активные списки**.

Просмотр активных списков

Чтобы просмотреть активный список:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит страницу с карточкой этого активного списка с подробной информацией о нем.

В верхней части карточки отображается схема активного списка.

Чтобы закрыть карточку и вернуться к списку активных списков, нажмите на кнопку **Назад** в левом верхнем углу карточки.

Панель инструментов карточки включает в себя следующие компоненты:

- Кнопка **Редактировать** ( ) позволяет изменить настройки активного списка.
- Кнопка **Удалить** ( ) позволяет удалить активный список.
- ID и описание активного списка.

В рабочей области карточки отображается таблица имеющихся в активном списке записей. При выборе конкретной записи в правой части рабочей области отображается ее карточка с детальной информацией.

В верхней части карточки записи имеются кнопки для редактирования ( ) и удаления ( ) записи.

В карточке отображается список полей записи.

Над таблицей записей расположены следующие инструменты для управления записями:

- Кнопка **Добавить запись** ( ) позволяет добавить новую запись в активный список.
- Кнопки **Экспорт** ( ) и **Импорт** ( ) позволяют выгружать существующие записи из активного списка и загружать в активный список записи извне.
- Поле **Поиск** предназначено для быстрого поиска записей в активном списке по ключу.

Таблица содержит следующие столбцы:

- **Ключ** — ключ записи в активном списке.
- **Повторы** — количество повторов записи в активном списке.
- **Срок действия** — дата и время прекращения действия записи в активном списке.
- **Дата создания** — дата и время создания записи в активном списке.
- **Дата изменения** — дата и время последнего изменения записи в активном списке.

В таблице также присутствуют столбцы, соответствующие полям активного списка. Поля ключа обозначены иконкой в заголовках столбцов.

При работе с таблицей доступны следующие операции:

- Поиск записей по полю **Ключ**.
- Настройка отображения таблицы.

Изменение активных списков

Чтобы изменить активный список:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит страницу с карточкой этого активного списка с подробной информацией о нем.
3. Нажмите на кнопку **Редактировать** () на панели инструментов карточки активного списка. Отобразится окно изменения активного списка.
4. Внесите изменения в требуемые поля активного списка.

> Схема активного списка выбирается только при создании активного списка. Изменение выбора схемы при редактировании активного списка недоступно.

5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление.

Удаление активных списков

Удаление недоступно для активных списков, используемых в других сущностях системы, то есть для списков, у которых в поле **Ресурсы** отображается значение, отличное от "0".

Чтобы удалить активный список:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит страницу с карточкой этого активного списка с подробной информацией о нем.
3. Нажмите на кнопку **Удалить** () на панели инструментов карточки активного списка. Отобразится окно подтверждения удаления активного списка.

4. Нажмите на кнопку **Удалить**. Система удалит активный список и отобразит соответствующее уведомление. Удаленный активный список будет исключен из перечня списков.

Вместе с активным списком также удаляются и все добавленные в него записи.

Поиск записей активного списка

Искать по активным спискам можно в разделе **Поиск**.

Чтобы просмотреть информацию о записи, нажмите на строку с ней в списке. В правой части экрана отобразится карточка записи.

Работа с записями активного списка

Доступные операции над записями активного списка:

- Добавление записей
- Просмотр записей
- Изменение записей
- Удаление записей
- Импорт записей
- Экспорт записей

Добавление записей

Чтобы добавить новую запись в активный список:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит страницу с карточкой этого активного списка с подробной информацией о нем.
3. Нажмите на кнопку **Добавить запись** (+) над списком записей активного списка. Отобразится окно добавления записи.
4. Заполните значения полей ключа и полей данных записи.
5. Нажмите на кнопку **Добавить**. Система добавит новую запись в активный список и отобразит соответствующее уведомление. Запись отобразится в перечне записей в карточке активного списка.

Просмотр записей

Чтобы просмотреть запись в активном списке:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит страницу с карточкой этого активного списка с подробной информацией о нем.

3. Нажмите на строку записи в активном списке. Система отобразит в правой части рабочей области карточку этой записи.

В верхней части карточки отображается ключ записи.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

В карточке отображается информация о выбранной записи, сгруппированная по следующим секциям:

- **Поля записи** — сведения о полях ключа и полях данных записи. Поля ключа обозначены иконкой .
- **Сведения** — общие сведения о записи:
  - **Повторы** — количество повторов записи.
  - **Срок действия** — дата и время прекращения действия записи.
  - **Дата создания** — дата и время создания записи.
  - **Дата изменения** — дата и время последнего изменения записи.

Изменение записей

Чтобы изменить запись в активном списке:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит страницу с карточкой этого активного списка с подробной информацией о нем.
3. Нажмите на строку записи в активном списке. Система отобразит в правой части рабочей области карточку этой записи.
4. Нажмите на кнопку **Редактировать** () в верхней части карточки. Отобразится окно изменения записи.
5. Внесите изменения в требуемые поля записи.

> Поля ключа записи задаются только при добавлении записи в активный список. Изменение значений полей ключа при редактировании записи недоступно.

6. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление.

Удаление записей

Чтобы удалить запись из активного списка:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит страницу с карточкой этого активного списка с подробной информацией о нем.

3. Нажмите на строку записи в активном списке. Система отобразит в правой части рабочей области карточку этой записи.
4. Нажмите на кнопку **Удалить** ( ) в верхней части карточки. Отобразится окно подтверждения удаления записи.
5. Нажмите на кнопку **Удалить**. Система удалит запись из активного списка и отобразит соответствующее уведомление. Удаленная запись будет исключена из перечня записей в карточке активного списка.

Импорт записей

Чтобы импортировать записи в активный список:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит страницу с карточкой этого активного списка с подробной информацией о нем.
3. Нажмите на кнопку **Импорт** ( ) над списком записей. Система отобразит окно импорта записей в активный список.
4. Нажмите **Выбрать файл** и выберите файл в формате CSV для импорта.
5. Нажмите на кнопку **Импортировать**. Система загрузит записи в активный список и отобразит соответствующее уведомление. Записи отобразятся в перечне записей в карточке активного списка.

Импортированные записи могут отобразиться с задержкой, поскольку операции вставки в активный список выполняются асинхронно и могут быть завершены после уведомления об успешном импорте.

Экспорт записей

Чтобы экспортировать записи из активного списка:

1. Перейдите в раздел **Ресурсы** → **Активные списки**. Система отобразит сведения об имеющихся активных списках.
2. Нажмите на строку активного списка в списке. Система отобразит страницу с карточкой этого активного списка с подробной информацией о нем.
3. Нажмите на кнопку **Экспорт** ( ) над списком записей. Система запросит подтверждение выполнения экспорта.
4. Нажмите **Экспортировать**. Система загрузит записи активных списков на устройство пользователя в формате CSV и отобразит соответствующее уведомление.

Работа с секретами

Данный раздел описывает процесс работы с секретами в системе. Работа осуществляется в разделе **Ресурсы** → **Секреты** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют

соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О секретах

Секрет — это компонент системы, который обеспечивает безопасное хранение чувствительных данных, таких как учетные данные, токены, строки подключения, сертификаты, приватные ключи. Секреты позволяют ссылаться на эти данные и использовать их без явного указания в различных объектах системы.

Секреты можно использовать в конфигурации конвейера для подключения к источникам данных в точках входа и/или внешним системам в конечных точках.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новый секрет.
- Поле **Поиск** предназначено для быстрого поиска секретов в отображаемом списке по названию.
- Кнопка **Фильтр** (▽) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка секретов.

В рабочей области отображается таблица имеющихся секретов. Таблица содержит следующие столбцы:

- **Название** — имя секрета, используемое для его идентификации в системе.
- **Описание** — краткое описание функциональности секрета.
- **Тип** — тип информации, хранящейся в секрете.
- **Статус** — текущий статус секрета (включен или выключен).
- **Дата создания** — дата и время создания секрета.
- **Дата изменения** — дата и время последнего изменения секрета.

При работе с таблицей секретов доступны следующие операции:

- Поиск секретов по полю **Название**.
- Фильтрация секретов по полям **Статус** и **Тип**. Статус и тип выбираются из выпадающих списков.
- Сортировка секретов по полям **Название**, **Дата создания** и **Дата изменения**.
- Настройка отображения таблицы.

При выборе конкретного секрета в правой части рабочей области отображается его карточка с детальной информацией, кнопкой ⋮ и переключателем статуса секрета.

Кнопка ⋮ расположена в верхнем правом углу карточки. При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранного секрета: изменение и удаление.

Работа с секретом

Доступные операции над секретами:

- Создание секрета
- Просмотр секрета
- Включение секрета
- Изменение секрета
- Удаление секрета

Создание секрета

Чтобы создать секрет:

1. Перейдите в раздел **Ресурсы** → **Секреты**. Система отобразит сведения об имеющихся секретах.
2. Нажмите на кнопку **Создать** (┼). Отобразится окно создания секрета.
3. Введите название секрета.
4. При необходимости введите описание секрета.
5. Выберите из выпадающего списка **Тип секрета** тип информации, которая будет храниться в секрете, и задайте настройки выбранного типа:
   - **Учетные данные** — укажите имя пользователя и пароль.
   - **Токен** — укажите токен.
   - **Строка подключения** — введите строку подключения.

     Строка подключения используется в точках входа AMQP и Database. Формат строки отличается в зависимости от типа и настроек точки.

   - **Сертификат** — загрузите файл сертификата. Допустимые форматы: DER, PEM (X.509).
   - **Контейнер PKCS** — контейнер представляет собой архив, в котором хранятся файл сертификата и файл приватного ключа. Чтобы настроить контейнер PKCS:
     - Загрузите файл сертификата. Допустимые форматы: DER, PEM (X.509).
     - Загрузите файл приватного ключа. Допустимые форматы: DER, PEM (PKCS#8).
     - При необходимости укажите пароль для приватного ключа.
   - **Контейнер PKCS#12** — контейнер представляет собой архив, в котором хранится файл сертификата с уже включенным в него приватным ключом. Чтобы настроить контейнер PKCS#12:
     - Загрузите файл сертификата. Допустимый формат: PKCS#12.
     - При необходимости укажите пароль для приватного ключа.
   - **NATS Key** — секрет предназначен для подключения к брокеру сообщений NATS с помощью приватного и публичного ключей. Чтобы настроить NATS Key, введите приватный ключ NKey Seed и публичный ключ NKey User.
6. Нажмите на кнопку **Создать**. Система создаст секрет и отобразит соответствующее

уведомление. Новый секрет появится в списке раздела **Ресурсы** → **Секреты**.

Просмотр секрета

Чтобы просмотреть секрет:

1. Перейдите в раздел **Ресурсы** → **Секреты**. Система отобразит сведения об имеющихся секретах.
2. Нажмите на строку секрета в списке. Система отобразит в правой части экрана карточку этого секрета с подробной информацией о нем.

В верхней части карточки отображается название секрета.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Карточка содержит следующие поля:

- **ID** — уникальный идентификационный код секрета, генерируемый автоматически при его создании.
- **Название** — имя секрета, используемое для его идентификации в системе.
- **ID тенанта** — уникальный идентификационный код тенанта, к которому относится секрет.
- **Статус** — текущий статус секрета (включен/выключен).
- **Описание** — краткое описание функциональности секрета.
- **Тип секрета** — тип информации, хранящейся в секрете.
- **Дата создания** — дата и время создания секрета.
- **Создал** — пользователь, инициировавший создание секрета.
- **Дата изменения** — дата и время последнего изменения секрета.
- **Изменил** — пользователь, выполнивший последнее изменение секрета.

Можно экспортировать секрет из объектного хранилища NATS Object Store.

Включение секрета

Для того чтобы использовать секрет в системе, его нужно включить.

Чтобы включить секрет:

1. Перейдите в раздел **Ресурсы** → **Секреты**. Система отобразит сведения об имеющихся секретах.
2. Нажмите на строку секрета в списке. Система отобразит в правой части экрана карточку этого секрета с подробной информацией о нем.
3. Переведите переключатель состояния секрета в верхней части карточки в активное положение. Отобразится окно подтверждения включения секрета.
4. Нажмите на кнопку **Включить**. Система включит секрет и отобразит соответствующее уведомление. Секрет станет доступен для добавления в объекты системы.

Если при включении секрета возникли ошибки, система отобразит сообщение об этом. Для

просмотра списка ошибок нажмите на кнопку **Показать**.

Чтобы выключить активный секрет:

1. Перейдите в раздел **Ресурсы** → **Секреты**. Система отобразит сведения об имеющихся секретах.
2. Нажмите на строку секрета в списке. Система отобразит в правой части экрана карточку этого секрета с подробной информацией о нем.
3. Переведите переключатель состояния секрета в верхней части карточки в неактивное положение. Отобразится окно подтверждения выключения секрета.
4. Нажмите на кнопку **Выключить**. Система выключит секрет и отобразит соответствующее уведомление. Секрет перестанет быть доступным для добавления в объекты системы.

Объекты, использующие секрет, продолжат его использовать даже после выключения. Выключение секрета запрещает только его новое добавление в объекты системы.

Изменение секрета

Чтобы изменить секрет:

1. Перейдите в раздел **Ресурсы** → **Секреты**. Система отобразит сведения об имеющихся секретах.
2. Нажмите на строку секрета в списке. Система отобразит в правой части экрана карточку этого секрета с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек секрета.
4. Внесите изменения в требуемые поля секрета.
5. Нажмите на кнопку **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление. Обновленная информация отобразится в карточке и списке секретов.

Изменить название и описание секрета можно непосредственно из его карточки:

1. Перейдите в раздел **Ресурсы** → **Секреты**. Система отобразит сведения об имеющихся секретах.
2. Нажмите на строку секрета в списке. Система отобразит в правой части экрана карточку этого секрета с подробной информацией о нем.
3. Нажмите на кнопку рядом с текущим названием или описанием секрета в его карточке. Система отобразит поле ввода значения.
4. Введите новое значение поля.
5. Нажмите **Сохранить**. Обновленная информация отобразится в карточке и списке секретов.

Удаление секрета

Чтобы удалить секрет:

1. Перейдите в раздел **Ресурсы** → **Секреты**. Система отобразит сведения об имеющихся секретах.
2. Нажмите на строку секрета в списке. Система отобразит в правой части экрана карточку этого секрета с подробной информацией о нем.

3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления секрета.
4. Нажмите на кнопку **Удалить**. Система удалит секрет и отобразит соответствующее уведомление. Удаленный секрет будет исключен из списка.

Удаление недоступно для секретов, используемых на конвейерах.

Работа с доменами

Данный раздел описывает процесс работы с доменами в системе. Работа осуществляется в разделе **Ресурсы** → **Домены** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О доменах

Домен — основная структурная единица системы. Это моделируемая предметная область, содержащая определенные в домене сущности, их структуры и взаимосвязи.

Работа с доменами осуществляется в разделе **Ресурсы** → **Домены**.

Описание интерфейса

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Зарегистрировать** (＋) позволяет зарегистрировать новый домен.
- Поле **Поиск** предназначено для быстрого поиска доменов в отображаемом списке по названию.
- Выпадающий список **Статус** позволяет настроить отображение списка доменов в соответствии с выбранным статусом.

В рабочей области отображается таблица существующих в системе доменов. Колонки таблицы представлены следующим образом:

- **Название** — название домена (задается при регистрации домена).
- **Статус** — статус регистрации домена:
  - **Регистрация** — в настоящий момент система регистрирует домен в системе и производит валидацию схемы домена.
  - **Зарегистрирован** — регистрация домена в системе завершена успешно.
  - **Обновление** — в настоящий момент система обновляет версию схемы домена.
  - **Ошибка регистрации** — регистрация домена завершена с ошибкой.
  - **Ошибка обновления** — обновление домена завершено с ошибкой.
- **Версия схемы** — версия схемы домена, выбранной при создании домена.
- **Дата создания** — дата создания домена.

- **Дата изменения** — дата изменения домена.
- **Описание** — краткое описание домена (задается опционально при регистрации домена).

При работе с таблицей доменов доступны следующие операции:

- Поиск доменов по полю **Название**.
- Фильтрация доменов по полю **Статус**. Статус выбирается из выпадающего списка.
- Сортировка доменов по полям **Название**, **Дата создания** и **Дата изменения**.

Работа с доменом

Доступные операции над доменом:

- регистрация домена;
- просмотр домена;
- изменение домена;
- обновление домена;
- удаление домена.

Регистрация домена

Чтобы добавить домен:

1. Перейдите в раздел **Ресурсы** → **Домены**. Система отобразит сведения об имеющихся доменах.
2. Нажмите на кнопку **Зарегистрировать** (＋). Отобразится окно регистрации домена.
3. Введите название домена.
4. При необходимости введите описание домена.
5. Выберите из выпадающего списка схему домена, на основе которой будет создан домен.

   В этом списке отображаются только последние версии опубликованных схем доменов со статусом **Включен**, по которым не зарегистрированы домены.

6. Нажмите на кнопку **Зарегистрировать**. Домен отобразится в таблице раздела **Домены** со статусом **Регистрация**. Система начнет процесс регистрации домена, а затем отобразит уведомление об успешной регистрации.

   В процессе регистрации домена система проводит валидацию схемы домена. Если процесс регистрации завершился с ошибкой, домену присваивается статус **Ошибка регистрации**, а в карточке домена отображаются следующие кнопки:

   - кнопка **Детали** — просмотреть подробную информацию об ошибке. (Ознакомьтесь со списком возможных ошибок при валидации.)
   - кнопка **Запустить регистрацию заново** — повторно запустить процесс регистрации домена.

Просмотр домена

Зарегистрированный домен отображается:

- В списке доменов раздела **Домены**. Здесь происходит управление доменом.
- В списке разделов в главном меню системы. Здесь происходит непосредственная работа с сущностями домена. В разделе домена в меню отображаются разделы сущностей, определенных схемой домена.

При выборе в главном меню какого-либо раздела сущности домена система отобразит в центральной части окна таблицу со списком экземпляров этой сущности. Если в схеме домена определено компактное представление информации, при выборе конкретной записи в правой части рабочей области отображается компактное представление экземпляра сущности.

Чтобы просмотреть домен:

1. Перейдите в раздел **Ресурсы** → **Домены**. Система отобразит сведения об имеющихся доменах.
2. Нажмите на строку домена в списке. Система отобразит в правой части экрана карточку этого домена с подробной информацией о нем. Название домена отображается в верхней части карточки.

Чтобы закрыть карточку, нажмите на крестик в правом верхнем углу карточки.

По умолчанию при выборе домена открывается вкладка **Информация**. На вкладке отображается общая информация о выбранном домене.

На вкладке **Информация** отображаются следующие поля записи:

- Идентификатор домена.
- Название домена.
- Описание домена.
- Статус регистрации домена.
- Информация о схеме, по которой зарегистрирован домен:
  - активная ссылка для перехода к просмотру схемы домена.
  - идентификатор схемы.
  - версия схемы, по которой зарегистрирован домен.
  - автор схемы.
  - теги схемы.
- Дата регистрации домена.
- Идентификатор пользователя, зарегистрировавшего домен.
- Дата изменения домена.
- Идентификатор пользователя, изменившего домен.

Чтобы просмотреть более подробную информацию о процессе регистрации или обновления домена, который завершился с ошибкой, нажмите на кнопку **Детали** рядом с полем **Статус** в карточке

домена. Система отобразит подробную информацию в новом окне.

Изменение домена

Чтобы изменить домен:

1. Перейдите в раздел **Ресурсы** → **Домены**. Система отобразит сведения об имеющихся доменах.
2. Нажмите на строку домена в списке. Система отобразит в правой части экрана карточку этого домена с подробной информацией о нем.
3. Выберите опцию **Изменить** в выпадающем меню **Действия** в верхней части карточки домена. Отобразится окно изменения полей настроек домена.
4. Внесите изменения в поля настроек домена.
5. Нажмите на кнопку **Сохранить**. Измененные данные домена будут сохранены. Система отобразит уведомление об изменении данных.

Изменить название и описание домена можно также непосредственно из карточки домена:

1. Перейдите в раздел **Ресурсы** → **Домены**. Система отобразит сведения об имеющихся доменах.
2. Нажмите на строку домена в списке. Система отобразит в правой части экрана карточку этого домена с подробной информацией о нем.
3. Нажмите на кнопку **Редактировать** ( ) рядом с названием или описанием домена в его карточке. Система отобразит поле ввода значения.
4. Введите новое значение поля.
5. Нажмите **Сохранить**. Измененные данные домена будут сохранены. Система отобразит уведомление об изменении данных.

Обновление домена

Вы можете обновить домен только в случае, если в системе была опубликована более новая версия схемы домена, чем его текущая версия. Для обновления домена нужно изменить версию используемой в нем схемы на более новую.

Система отслеживает публикацию новых версий зарегистрированных доменов. После публикации новой версии схемы домена система:

- отображает для записи этого домена значок информации в столбце **Версия схемы** таблицы доменов.
- отображает в карточке домена сообщение о выходе новой версии схемы домена.
- делает активной команду **Обновить** в выпадающем меню **Действия** в верхней части карточки домена.

Домены со статусами **Регистрация**, **Обновление**, **Ошибка регистрации**, **Ошибка обновления** обновлять нельзя.

Чтобы обновить домен:

1. Перейдите в раздел **Ресурсы** → **Домены**. Система отобразит сведения об имеющихся доменах.

2. Нажмите на строку домена в списке. Система отобразит в правой части экрана карточку этого домена с подробной информацией о нем.
3. Выберите опцию **Обновить** в выпадающем меню **Действия** в верхней части карточки домена. Отобразится окно обновления домена.
    - в поле **Новая версия схемы** отображается номер последней опубликованной версии схемы домена.
    - В поле **Текущая версия схемы** отображается номер текущей версии схемы домена.
4. Нажмите на кнопку **Проверить**, чтобы запустить процесс проверки возможной потери данных при переходе на последнюю опубликованную версию схемы домена.

    В системе предусмотрена проверка возможной потери данных при обновлении домена. Система сверяет сущности в текущей и новой версиях схемы домена и выявляет наличие удаленных или переименованных сущностей. По итогам проверки в поле **Проверка потери данных** окна обновления отобразится информация об обнаруженных изменениях сущностей или их атрибутов, которые могут привести к потере данных. Если существенных изменений в новой версии схемы обнаружено не будет, система также сообщит об этом.

5. Нажмите на кнопку **Обновить**. Система начнет процесс обновления домена на более новую версию схемы. Статус домена в таблице во время процесса обновления изменится на **Обновление**. После обновления система отобразит соответствующее уведомление.

    В процессе обновления система проводит валидацию схемы домена. Если процесс обновления завершился с ошибкой, в карточке домена станут доступными следующие кнопки:

    - Кнопка **Детали** позволяет просмотреть подробную информацию об ошибке. (Ознакомьтесь со списком возможных ошибок валидации и с рекомендациями по их устранению.)
    - Кнопка **Запустить обновление заново** позволяет повторно запустить процесс обновления домена.

Удаление домена

Чтобы удалить домен:

1. Перейдите в раздел **Ресурсы** → **Домены**. Система отобразит сведения об имеющихся доменах.
2. Нажмите на строку домена в списке. Система отобразит в правой части экрана карточку этого домена с подробной информацией о нем.
3. Выберите опцию **Удалить** в выпадающем меню **Действия** в верхней части карточки домена. Отобразится окно удаления домена.
4. Нажмите на кнопку **Удалить**. Домен будет удален из системы. Система отобразит уведомление об удалении домена.

Вместе с доменом система также удаляет все связанные с ним сущности и атрибуты без возможности восстановления. Если в связанных доменах имелись ссылки на сущности удаленного домена, в связанных доменах могут возникать ошибки.

Интерфейс списка объектов в домене

После регистрации домена, для которого в схеме определен интерфейс списка объектов, в главном меню системы отображается раздел с названием домена. Подразделами домена могут быть сущности, определенные схемой домена. Список сущностей представлен в виде таблицы.

Если в схеме определен просмотр сущности, эта информация отображается при выборе экземпляра сущности в компактном представлении сущности в правой панели.

Списочное представление с контекстом

В списочном представлении доступно отображение экземпляров сущностей в рамках определенного контекста. Таким образом, при нажатии на строку объекта в представлении можно отображать другое списочное представление.

Переход к отображению другого представления может осуществляться по нажатию на любую кнопку, к которой привязано действие перехода к списочному представлению с контекстом. Действие может быть привязано к кнопкам на панели действий в списочном представлении, кнопкам или меню в компактном представлении. К экземплярам представления система применяет контекст — фильтр по определенному критерию, который задается в схеме домена.

▼ *Пример*

Просмотр списочного представления: **Оценка требований**

Контекст: **Аудит: на соответствие 307-ФЗ от 30.12.2008**

Возможность отображения раздела меню системы в левой панели, соответствующего представлению с контекстом, зависит от настроек в схеме домена.

Контекст можно определить для следующих видов списочного представления:

- без вкладок (одно представление сущности);
- с вкладками (несколько представлений сущности).

Контексты для списочного представления без вкладок

Если для списочного представления в схеме домена определены контексты, над областью фильтрации отображаются закрепленные ярлыки с описанием контекстов.

Название ярлыка формируется из условия контекста по следующему шаблону:

`"Название сущности в схеме домена: Лейбл"`. Лейбл — это комбинация атрибутов сущности для отображения в представлении.

Для примера выше: "**Аудит: на соответствие 307-ФЗ от 30.12.2008**".

Если название контекста не умещается на ярлыке, для его отображения наведите курсор мыши на ярлык.

Ярлык контекста можно удалить, нажав на крестик в его правой части и подтвердив действие. Если крестик отсутствует, ярлык недоступен для удаления согласно настройке в схеме домена.

При обновлении или повторном открытии страницы ярлык с контекстом снова отобразится на странице.

Особенности работы с контекстами для списочного представления с вкладками

В схеме домена можно настроить отображение нескольких списочных представлений для одной сущности на разных вкладках в рабочей области экрана. В этом случае контексты можно определить для каждой вкладки в списочном представлении сущности отдельно.

Фильтрация в представлении с контекстом

Экземпляры сущности в списочном представлении, которые отображаются в рамках контекста, можно отфильтровать по доступным критериям фильтрации, так же как и любые другие экземпляры сущности. Фильтрация применяется в рамках вкладки списочного представления. При удалении ярлыка контекста имеющийся фильтр по-прежнему применяется к экземплярам представления.

Переход к представлению с контекстом

Чтобы перейти к представлению с контекстом:

1. Нажмите в списочном представлении на строку экземпляра, связанного с представлением с контекстом. Компактное представление экземпляра отобразится в правой части экрана.
2. Нажмите на кнопку или иконку, инициирующую действие открытия, в компактном представлении. Система отобразит в отдельной вкладке браузера списочное представление с контекстом.

Фильтрация в списочном представлении

Вы можете добавить, применить, изменить, отключить или включить, инвертировать и удалить пользовательские фильтры экземпляров сущностей в представлении.

Добавление фильтра списка экземпляров сущностей

Чтобы добавить пользовательский фильтр экземпляров сущностей:

1. Нажмите на кнопку **Добавить фильтр**. Отобразится окно добавления фильтра.
2. Выберите из выпадающего списка поле, по которому будут фильтроваться экземпляры.
3. Выберите из выпадающего списка оператор, который будет применен к значению поля:
   - **=** — равно.
   - **!=** — не равно.
   - **In** — совпадение с одним из перечисленных значений. Значения указываются через запятую.
   - **Like** — соответствие заданному шаблону.
   - **>** — больше.
   - **<** — меньше.
   - **>=** — больше или равно.
   - **<=** — меньше или равно.

- **Between** — входит в заданный диапазон.

4. Укажите значение поля. Для ввода нескольких значений укажите их через запятую.
5. Если нужно инвертировать фильтр, включите переключатель **Инвертировать (NOT)**.
6. Если нужно указать название фильтра, включите переключатель **Настроить заголовок** и введите название.
7. Нажмите на кнопку **Добавить**. Созданный фильтр появится рядом с кнопкой **Добавить фильтр**.

Если задан хотя бы один фильтр, система отобразит кнопки:

- **Отключить все** — отключение всех фильтров.
- **Включить все** — включение всех фильтров.
- **Инвертировать (NOT) все** — инвертирование всех фильтров.
- **Удалить все** — удаление всех фильтров.

Изменение фильтра экземпляров сущностей

Чтобы изменить заданный пользовательский фильтр экземпляров сущностей:

1. Нажмите на стрелку слева от названия фильтра. Отобразится выпадающее меню действий под фильтром.
2. Выберите опцию **Изменить** в выпадающем меню. Откроется окно изменения фильтра.
3. Внесите изменения в требуемые поля.
4. Нажмите на кнопку **Сохранить**. Измененные настройки фильтра будут сохранены.

Инвертирование фильтра экземпляров сущностей

Чтобы инвертировать условие пользовательского фильтра экземпляров сущностей, выберите опцию **Инвертировать (NOT)** в выпадающем меню. Условие фильтра будет заменено на противоположное.

Отключение фильтра экземпляров сущностей

По умолчанию пользовательский фильтр создается включенным и учитывается при фильтрации экземпляров сущностей.

Чтобы отключить фильтр, выберите опцию **Отключить** в выпадающем меню. Фильтр будет отключен и не будет использоваться для фильтрации.

Включение фильтра экземпляров сущностей

Чтобы включить отключенный пользовательский фильтр экземпляров сущностей, выберите опцию **Включить** в выпадающем меню. Фильтр будет включен и применен для фильтрации.

Удаление фильтра экземпляров сущностей

Чтобы удалить пользовательский фильтр экземпляров сущностей, выберите опцию **Удалить** в выпадающем меню. Фильтр будет сброшен без возможности восстановления.

Работа с фильтрами значений атрибута `Reference`

В качестве критерия в пользовательском фильтре экземпляров сущностей можно использовать значения атрибута типа `Reference` (ссылка на другую сущность).

Способ создания фильтра такой же, как и при работе с другими полями. В окне добавления фильтра:

1. Выберите из выпадающего списка поле компактного представления, в котором отображается ссылка.
2. Укажите требуемый оператор. Доступные операторы:
   - **=** — равно.
   - **!=** — не равно.
   - **In** — совпадение с перечисленными значениями.
3. Выберите значение поля (или несколько значений в зависимости от оператора) из выпадающего списка. Для выбора будут доступны экземпляры сущности, на которые существует ссылка.

Работа с фильтрами связей

В качестве критерия фильтрации можно использовать наличие связи с определенными объектами.

В окне добавления фильтра:

1. Выберите из выпадающего списка название связи (варьируется в зависимости от ее характеристик).
2. Укажите требуемый оператор. Доступные операторы:
   - **=** — равно.
   - **!=** — не равно.
   - **In** — совпадение с перечисленными значениями.
3. Выберите лейбл или лейблы (в зависимости от оператора) из раскрывающегося списка.

Система отобразит в списочном представлении экземпляры сущностей, связанные с объектами, указанными в фильтре.

В поле выбора связи практически во всех случаях отображается название связи для экземпляра сущности, компактное представление которой открыто. Исключением является симметричная направленная связь. В этом случае в списке для выбора отображаются названия для **обеих** сторон связи. См. примеры ниже.

▼ *Примеры отображения названия связи*

- Связь асимметричная, то есть связанные сущности разные.
  - Связь направленная. Отобразится название связи для стороны сущности, компактное представление которой открыто.

    Пример связи: **Уязвимость** *порождает* **Задачу**. В поле окна добавления фильтра отобразится название связи для стороны сущности **Уязвимость**, компактное представление которой открыто, то есть **"Порождает"**. Связь направлена на сущность **Задача**.

- Связь ненаправленная. Отобразится название связи для стороны сущности, компактное представление которой открыто.

  Пример связи: **Пользователь** *связан с* **Задачей**. В поле окна добавления фильтра отобразится название связи для стороны сущности **Пользователь**, компактное представление которого открыто, то есть "**Связан с**".

- Связь симметричная, то есть связанные сущности одинаковые.
  - Связь направленная. Отобразится название связи для первой и второй сторон связи.

    Пример связи: **Задача 1** блокирует **Задачу 2**. В поле окна добавления фильтра отобразится название связи для первой и второй сторон связи, то есть "**Блокирует**" и "**Блокируется**".

  - Связь ненаправленная. Отобразится название связи для первой стороны связи.

    Пример связи:

    Пользователь 1 работает в одной команде с **Пользователем 2**. В поле окна добавления фильтра отобразится название связи для первой стороны связи, т.е. "**Работает в одной команде с**".

Компактное представление сущности

Работа с таблицами

Если в схеме домена с помощью пользовательского атрибута `table` создана таблица для отображения в компактном представлении, название таблицы отображается в представлении над таблицей.

При нажатии на кнопку + в таблицу добавляется новая пустая строка. Правила редактирования полей в строке определяются типом данных атрибута в поле. Введенные значения сохраняются в системе по кнопке **Сохранить**.

Если введенное в поле значение не проходит валидацию, система подсвечивает поле красной рамкой и отображает сообщение об ошибке, определенное в схеме домена.

При наведении курсора на поле появляется кнопка , при нажатии на которую поле можно отредактировать.

При нажатии на кнопку ⋮ отображается меню с командами, позволяющими изменить и удалить всю строку.

Параметры фильтрации и пагинации, заданные в схеме домена, отображаются в таблице, соответственно, в заголовках колонок таблицы и под таблицей.

Управление тегами экземпляра

Если в компактном представлении настроено отображение поля **Теги**, в поле отображается набор тегов, связанных с экземпляром, облегчающий его поиск и классификацию.

Кнопка **Редактировать** в поле **Теги** экземпляра позволяет:

- удалить теги;
- добавить теги из выпадающего списка тегов сущности.

Вы также можете создать новые теги и добавить их в список тегов сущности.

Вкладка Действия в компактном представлении

Если в компактном представлении настроено отображение действия запуска скрипта или плейбука, на вкладке **Действия** можно просмотреть в таблице результаты запуска в отношении этого экземпляра.

Колонки таблицы представлены следующим образом:

- **Название** — название запускаемого скрипта или плейбука.
- **Время старта** — дата и время запуска.
- **Статус** — статус запуска.

  Доступные статусы:

  - **Выполнен**;
  - **В процессе**;
  - **Ошибка**.

Переход к связанному экземпляру сущности

Отображение атрибутов сущности в представлении задается в схеме домена с помощью виджетов атрибутов.

Если в компактном представлении сущности используется атрибут типа `Reference`, то становятся доступны следующие действия:

- Просмотр объекта, на который дана ссылка.

  Для этого нажмите левой кнопкой мыши на лейбл объекта. Система отобразит форму компактного представления для этого объекта.

  Вернуться к компактному представлению исходной сущности можно по кнопке **Назад** в заголовке компактного представления.

- Открытие ссылки в новой вкладке или окне.

  Для этого нажмите правой кнопкой мыши на лейбл объекта и в отобразившемся стандартном меню браузера выберите одно из следующих действий:

  - Открыть ссылку в новом окне.
  - Открыть ссылку в новой вкладке.

  Система отобразит форму списочного представления, в котором будет открыто компактное представление для связанного экземпляра.

История операций над экземпляром сущности

В схеме домена можно определить, что в компактном представлении должна отображаться история операций над экземпляром сущности. В этом случае в представлении сущности отображается вкладка **История** со списком таких операций (например, создание или редактирование экземпляра сущности).

Комментарии к экземпляру сущности

В схеме домена для списочного представления можно определить возможность комментировать экземпляр сущности. В этом случае в компактном представлении сущности в правой панели отображается вкладка **Комментарии**. Комментарии сортируются по дате создания комментария по убыванию.

Комментарии к экземпляру сущности можно добавлять, изменять и удалять.

Добавление комментария

Чтобы добавить комментарий:

1. Зайдите в зарегистрированный домен со списочным представлением.
2. Выберите экземпляр сущности.
3. Перейдите на вкладку **Комментарии** в компактном представлении сущности (в правой панели).
4. Нажмите на кнопку **Добавить**.
5. Введите данные в поле ввода комментария.
6. Нажмите кнопку **Отправить**. Комментарий к экземпляру сущности будет добавлен. Система отобразит уведомление о добавлении комментария.

Изменение комментария

Чтобы изменить комментарий:

1. Зайдите в зарегистрированный домен со списочным представлением.
2. Выберите экземпляр сущности.
3. Перейдите на вкладку **Комментарии** в компактном представлении сущности в правой панели.
4. Наведите курсор на комментарий, который необходимо изменить. Отобразится кнопка **Редактировать**.
5. Нажмите на кнопку **Редактировать**.
6. Внесите изменения в комментарий.
7. Нажмите на кнопку **Отправить**. Комментарий к экземпляру сущности будет изменен и отмечен меткой **Изменено**. Система отобразит уведомление об изменении комментария.

Удаление комментария

Чтобы удалить комментарий:

1. Зайдите в зарегистрированный домен со списочным представлением.

2. Выберите экземпляр сущности.
3. Перейдите на вкладку **Комментарии** в компактном представлении сущности (в правой панели).
4. Наведите курсор на комментарий, который необходимо удалить. Отобразится кнопка **Удалить**.
5. Нажмите на кнопку **Удалить**. Комментарий к экземпляру сущности будет удален. Система отобразит уведомление об удалении комментария.

Настройка отображения легенды виджета визуализации

По умолчанию в правой части виджета визуализации отображается легенда, на которой приведены цветовые обозначения и названия рядов данных на виджете.

Легенда не отображается для виджетов с типом визуализации **Таблица** или **Итоги**.

Чтобы скрыть легенду, нажмите на кнопку действий (⋮) в верхнем правом углу виджета и выберите опцию **Скрыть легенду**. Легенда перестанет отображаться на виджете, а диаграмма визуализации данных займет всю ширину виджета.

Чтобы отобразить скрытую легенду, нажмите на кнопку действий (⋮) в верхнем правом углу виджета и выберите опцию **Показать легенду**. В правой части виджета отобразится его легенда, а диаграмма визуализации отмасштабируется по оставшемуся на виджете пространству.

Редактор связей

Этот раздел предназначен для редактирования связей между сущностями.

Редактирование связей осуществляется для выбранной сущности в разделе **Ресурсы** → **Домены**.

Редактор связей — это компонент системы, позволяющий задавать связи для сущностей.

О связях

Отношения между сущностями в доменах определяются при помощи связей.

Связь задается в схеме домена отдельным массивом, где хранятся ссылки на связанные сущности и ряд специальных параметров.

Как и сущность, связь может обладать атрибутами.

В системе доступна **линейная** связь. Она позволяет строить горизонтальные отношения между сущностями, задавая ограничения на количество объектов с левой и правой стороны связи.

Связи для сущности добавляются в разделе сущности домена, созданного по настроенной схеме, в панели компактного представления сущности.

Типы линейной связи

- **1-1** — позволяет связать сущность с другой сущностью по принципу "один — к одному". Сущности могут быть связаны этим типом связи только друг с другом.

  ▼ *Пример связи (**1-1**)*

Пользователь → ИД

У сущности **Пользователь** может быть только один уникальный идентификационный номер, и наоборот.

- **1-N** (или **N-1**) — позволяет связать одну сущность с несколькими сущностями по принципу "один — ко многим".

▼ *Пример связи (**N-1**)*

Учетная запись → Пользователь

У **Пользователя** может быть много учетных записей, но у конкретной **Учетной записи** всегда есть только один **Пользователь**.

- **N-N** — позволяет построить связи между многими сущностями по принципу "многие — ко многим". Несколько сущностей могут быть связаны этим типом связи с несколькими сущностями.

▼ *Пример связи (**N-N**)*

Сервер → Администратор Сущность **Сервер** можно привязывать к нескольким администраторам. Администратор* также может быть связан с несколькими серверами.

В схеме домена мы всегда связываем одну сущность с другой. При этом в доменах, созданных на основе схемы, связываются объекты (экземпляры сущностей). Таким образом, в связях 1-1 связываются два объекта, а в связях 1-N и N-N один или несколько объектов связываются с несколькими объектами.

Симметричность связи

Симметричная связь — в левой и правой частях связи указана одна и та же сущность.

▼ *Пример*

Линейная направленная связь типа N-N между сущностями **Задача** и **Задача**. Здесь Задача блокирует Задачу.

Асимметричная связь — в левой и правой частях связи указаны разные сущности.

▼ *Пример*

Линейная направленная связь типа 1-N между сущностями **Пользователь** и **Задача**. Здесь Пользователь является исполнителем по Задаче.

Направленность связи

- Направленная связь — имеет строго заданное направление от левой стороны связи к правой, определяя связанные с этим ограничения.

▼ *Пример направленной связи.*

В системе имеются сущности **Уязвимость** и **Задача**. Их соединяет направленная связь. Действие направлено от **Уязвимости** к **Задаче**. В результате **Уязвимость** породила **Задачу** (на устранение уязвимости).

- Ненаправленная связь — в отличие от направленной, не имеет строго заданного направления и возможна только для линейных связей типа N-N.

  ▼ *Пример ненаправленной связи.*

  В системе имеются сущности **Оборудование 1** и **Оборудование 2**. Их соединяет связь, но направление при этом отсутствует.

Название связи

Название связи отображается в интерфейсе и отражает характер отношения между сторонами связи. Задается отдельно для каждой из сторон связи.

▼ *Пример*

Линейная направленная связь типа 1-N между сущностями **Пользователь** и **Оборудование**:

- название связи, отображаемое со стороны Пользователя — **Владеет**;
- название связи, отображаемое со стороны Оборудования — **Владелец**.

Параметры связи

Для того чтобы к сущностям домена можно было добавлять связь, при настройке схемы домена необходимо указать следующие параметры связи:

- ID связи;
- Название связи (опционально);
- Описание связи, отображаемое в схеме домена (опционально).
- Ссылки на сущности для сторон связи слева и справа;
- Тип связи (1-1, 1-N, N-N);
- Название связи для каждой стороны. Например, инцидент "блокирует задачу", задача "блокируется инцидентом".
- Направление. Для связи типа N-N можно указать, является ли связь ненаправленной. По умолчанию связь направленная.
- Атрибуты связи. Атрибуты связи задаются так же, как и атрибуты сущностей.

Работа со связями

В схеме домена можно настроить отображение в компактном представлении сущности одного из следующих типов редактора связей:

- компактный редактор связей. Он задается для определенной стороны определенной связи. Для каждой связи отображается отдельное поле. Эти поля отображаются в списке в окне компактного представления сущности. Если для объекта уже выбраны объекты для связывания, их лейблы отображаются в поле связи. Лейбл — это комбинация атрибутов сущности для отображения в представлении.
- расширенный редактор связей. Он включает в себя таблицы связей для объектов, связанных с текущим объектом. Для каждой связи отображается отдельная таблица. Таблицы связи можно сворачивать, нажав на стрелку рядом с их названиями.

Расширенный редактор связей отображается только для экземпляров, которые можно связать с **несколькими** экземплярами сущности.

Вы можете настроить в схеме домена отображение как атрибутов связанных объектов, так и атрибутов связи. Для этого используется поле **columns** в настройках расширенного редактора связей. В нем нужно указать ID атрибутов связанных объектов или связи для отображения.

ID атрибутов связанных объектов и связи не должны совпадать. В противном случае система отобразит в представлении значение для атрибута сущности, так как приоритет такого атрибута будет выше.

Доступные типы редакторов для сторон типов связи:

- Для стороны N — **компактный** редактор (с выбором нескольких связанных экземпляров) и **расширенный** редактор.

  Если для одной и той же стороны заданы несколько редакторов, нужные редакторы можно использовать в разных представлениях сущности.

- Для стороны 1 — только **компактный** редактор (с выбором одного связанного экземпляра).

В редакторах можно выполнять определенные действия, описанные ниже:

- создание связи;
- удаление связи;
- просмотр связанного экземпляра сущности.

Создание связи

Создать связь, т.е. указать связанный экземпляр сущности, можно как в момент создания экземпляра, так и при работе с экземпляром, уже имеющимся в системе.

Система отображает доступные для связывания сущности в соответствии со следующими правилами:

- Если выбран Основной тенант, в интерфейсе системы отображаются все объекты справочников и организаций. Если выбран какой-либо дочерний тенант, отображаются только объекты, соответствующие этому тенанту.
- Текущий экземпляр сущности не отображается, так как связать экземпляр с ним же самим нельзя.
- Отображаются только экземпляры сущности, подходящие для выбранного типа связи и не связанные с текущей сущностью.

Создание связи в компактном редакторе

Чтобы создать связь:

- Для имеющегося экземпляра сущности:

a. Выберите экземпляр сущности в списочном представлении. Система отобразит компактное представление сущности в правой панели.

b. Добавьте связь из поля связи:

   i. Выберите в компактном представлении из выпадающего меню **Действия** (⋮) опцию **Редактировать**. Отобразится окно изменения сущности.

   ii. Нажмите на стрелку в поле с названием связи. В выпадающем списке отобразятся допустимые для связывания объекты.

   iii. Выберите сущность для связывания и нажмите кнопку **Сохранить**.

Вы также можете начать вводить в поле поиска списка последовательность символов. Система будет подставлять подходящие значения автоматически. Атрибуты сущности, по которым осуществляется поиск, настраиваются в схеме домена.

Система свяжет выбранную сущность с текущей сущностью и отобразит в поле связи.

Вместо использования меню **Действия** вы можете нажать на кнопку **Редактировать** (✎) рядом с полем связи и выбрать сущность для связи непосредственно в этом поле.

- Для нового экземпляра сущности:

a. Нажмите на кнопку ＋ (**Создать**) в списочном представлении сущности. Система отобразит окно создания экземпляра сущности.

b. Нажмите на стрелку в поле с названием связи. В выпадающем списке отобразятся допустимые для связывания объекты.

c. Выберите экземпляр сущности для связывания.

d. Заполните поля в окне создания и нажмите кнопку **Создать**.

Вы также можете начать вводить в поле поиска списка последовательность символов. Система будет подставлять подходящие значения автоматически. Атрибуты сущности, по которым осуществляется поиск, настраиваются в схеме домена.

Система свяжет выбранную сущность с текущей сущностью и отобразит в поле связи.

Создание связи в расширенном редакторе

Чтобы создать связь:

- Для имеющегося экземпляра сущности:

a. Выберите экземпляр сущности в списочном представлении. Система отобразит компактное представление сущности в правой панели. Отобразится таблица расширенного редактора связей.

b. Нажмите на кнопку ＋(**Создать**), расположенную справа сверху над таблицей редактора. Система отобразит таблицу добавления связей со списком всех экземпляров сущности,

соответствующих характеристикам связи.

c. Выберите экземпляры сущности, которые нужно связать с текущим экземпляром, установив флажки слева от их названий. Набор колонок этой таблицы задается в схеме домена.

d. Нажмите на кнопку **Создать**.

- Для нового экземпляра сущности:

a. Нажмите на кнопку + (**Создать**) в списочном представлении сущности. Система отобразит окно создания объекта. В нем отобразится таблица расширенного редактора связей.

b. Выберите экземпляры сущности, которые нужно связать с текущим экземпляром, установив флажки слева от их названий. Набор колонок этой таблицы задается в схеме домена.

c. Нажмите на кнопку **Создать**.

Если для экземпляра еще не заданы связи, вы также можете воспользоваться кнопкой **Создать**, отображаемой в таблице.

Просмотр связанного экземпляра сущности

Просмотр связанного экземпляра сущности в компактном редакторе

Чтобы просмотреть связанный экземпляр, выберите экземпляр сущности.

В компактном представлении сущности система отобразит в редакторе связей информацию о связанных с ней экземплярах (при их наличии).

Информацию об экземплярах сущностей, связанных с текущим экземпляром сущности, можно просмотреть из редактора связей. В текущей вкладке можно просмотреть компактное представление связанного экземпляра, а в новой вкладке — его списочное представление.

- Чтобы просмотреть выбранный связанный экземпляр, выполните одно из следующих действий (доступность действий определяется настройками схемы домена):
  - Нажмите левой кнопкой мыши на связанный экземпляр сущности. Система отобразит форму компактного представления для **связанного** экземпляра. Вернуться к компактному представлению исходной сущности можно по кнопке **Назад** в заголовке компактного представления.
  - Нажмите правой кнопкой мыши на связанный экземпляр сущности и в отобразившемся стандартном меню браузера выберите одно из следующих действий:
    - **Открыть ссылку в новом окне**;
    - **Открыть ссылку в новой вкладке**.

    Система отобразит форму списочного представления, в котором будет открыто компактное представление для связанного экземпляра.

Если для связанного экземпляра сущности не определено списочное представление, действия в меню браузера **Открыть ссылку в новой окне** и **Открыть ссылку в новом вкладке** отображены не будут.

Просмотр связанного экземпляра сущности в расширенном редакторе

Чтобы просмотреть связанный экземпляр, выберите экземпляр сущности.

В компактном представлении отобразятся таблицы для каждой из связей текущего экземпляра сущности. Они содержат связи текущего экземпляра сущности с другими экземплярами (при их наличии). Для каждой таблицы набор колонок задается в схеме домена. В них могут отображаться атрибуты связанных сущностей или атрибуты самой связи.

Выбранный связанный экземпляр текущего экземпляра сущности можно просмотреть в текущей или в новой вкладке. По нажатию на связанный экземпляр система отобразит меню, в котором можно выбрать одно из следующих действий для просмотра этой связи (доступность действий определяется настройками схемы домена):

- **Открыть** ( ) — открыть связанный экземпляр для просмотра в текущей вкладке.
- **Открыть в новой вкладке** ( ) — открыть связанный экземпляр для просмотра в новой вкладке. Действие недоступно, если у связанного объекта отсутствует списочное представление.

Удаление связи

Удаление связи в компактном редакторе

Чтобы удалить для текущего экземпляра связь с другим экземпляром:

1. Выберите экземпляр сущности. Система отобразит компактное представление сущности с полями связей в правой панели.
2. Удалить связь можно одним из следующих способов:
   - Из поля связи:
     i. Нажмите на кнопку рядом с названием связи. Поле связи станет доступным для редактирования.
     ii. В поле связи нажмите на крестик рядом с названием экземпляра, связь с которым нужно удалить.
     iii. Нажмите на кнопку **Сохранить**. Система удалит связь.
   - Из окна редактирования экземпляра сущности:
     i. Выберите в компактном представлении из выпадающего меню **Действия** ( ) опцию **Редактировать**. Отобразится окно редактирования экземпляра сущности.
     ii. В поле связи нажмите на крестик рядом с названием экземпляра, связь с которым нужно удалить.
     iii. Нажмите на кнопку **Сохранить**. Система удалит связь.

Внимание! Связь удаляется без отображения окна с подтверждением удаления.

Удаление связи в расширенном редакторе

Чтобы удалить для текущего экземпляра сущности связь с другим экземпляром:

1. Выберите экземпляр сущности и перейдите в режим редактирования. В окне с компактным

представлением отобразятся таблицы расширенном редактора связей для выбранного экземпляра. В верхней части таблицы отобразятся экземпляры, связанные с текущим экземпляром.

2. Снимите флажки для экземпляров, связь с которыми нужно удалить.
3. Нажмите на кнопку **Сохранить**. Система удалит выбранную связь с текущим экземпляром.

Внимание! Связь удаляется без отображения окна с подтверждением удаления.

Работа с переменными

Данный раздел описывает процесс работы с переменными в системе. Работа осуществляется в разделе **Ресурсы** → **Переменные** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О переменных

В скриптах и плейбуках, выполняемых системой, используются переменные и секреты.

Секрет — разновидность переменных, которая обеспечивает безопасное хранение чувствительных данных. Секреты позволяют ссылаться на эти данные и использовать их без явного указания в различных объектах системы. Значения секрета, в отличие от переменных, невозможно посмотреть. Они могут быть только перезаписаны. Доступ к секретам является событием аудита и регистрируется системой.

Использование переменных в скрипте

Переменные в скриптах и плейбуках можно использовать как в исходном коде, так и в качестве входных аргументов, задаваемых в блоке **Аргументы** в параметрах. При добавлении в систему скриптам назначается уникальный идентификатор формата UUID, однако для удобства рекомендуется использовать уникальное имя.

Поддерживаются следующие типы переменных:

- учетные данные;
- токен;
- строка подключения;
- сертификат;
- контейнер PKCS;
- контейнер PKCS#12.

Переменные и секреты хранятся в зашифрованном виде. При добавлении переменных и секретов в поля конфигурации, в полях отображаются только их имена. Сами значения подставляются только в момент запуска скрипта с этой конфигурацией. Значение секрета, в отличие от значения переменной, невозможно просмотреть. Оно может быть только перезаписано.

Событие доступа к секретам идентифицируется как отдельное событие аудита. Затем оно регистрируется и сохраняется в системное хранилище аудита.

Интерфейс раздела

Панель инструментов включает в себя следующие компоненты:

- Кнопка **Создать** (＋) позволяет создать новую переменную.
- Поле **Поиск** предназначено для быстрого поиска переменных по названию в отображаемом списке.
- Кнопка **Фильтр** позволяет настроить следующие критерии фильтрации переменных: по **Статусу**, наличию признака **Секрет** и **Типу** искомых элементов.

В рабочей области отображается таблица имеющихся переменных. При выборе конкретной переменной на экране открывается карточка с детальной информацией о ней.

Колонки таблицы представлены следующим образом:

- **Название** — название переменной.
- **Описание** — описание переменной.
- **Секрет** — индикатор, указывающий, является ли переменная секретом.
- **Тип** — тип переменной.
- **Статус** — статус переменной (**Включен**, **Выключен**).
- **Дата создания** — дата создания переменной.
- **Дата изменения** — дата и время изменения переменной.

При работе с таблицей переменных доступны следующие операции:

- Поиск переменных по полю **Название**.
- Фильтрация переменных по полям **Статус**, **Секрет**, **Тип**. Статус, индикатор секрета и тип переменной выбираются из выпадающих списков.
- Сортировка переменных по полям **Название**, **Секрет**, **Дата создания** и **Дата изменения**.

Работа с переменной

Доступные операции над переменной:

- создание переменной;
- просмотр переменной;
- изменение переменной;
- удаление переменной.

Создание переменной

Чтобы создать переменную:

1. Перейдите в раздел **Ресурсы** → **Переменные**.
2. Нажмите на кнопку **Создать** (➕). Отобразится окно создания переменной.
3. Введите уникальный идентификатор переменной. Рекомендуется использовать человекочитаемое значение.
4. Введите название переменной.
5. При необходимости введите описание переменной.
6. Выберите из выпадающего списка тип переменной. Отобразятся необходимые поля для ввода значения переменной.
7. Если переменная является секретом, переведите переключатель **Секрет** в активное положение.
8. Введите значение переменной.

   При выборе типов переменных, приведенных ниже, переключатель **Секрет** устанавливается в активное положение по умолчанию. Укажите требуемые данные в полях, отображаемых системой:

   - Учетные данные. Введите имя пользователя и пароль.
   - Токен. Введите токен.
   - Строка подключения. Введите строку подключения.
   - Сертификат. Загрузите файл сертификата. Допустимые форматы: DER, PEM (X.509).
   - Контейнер PKCS.
     - Загрузите файл сертификата. Допустимые форматы: DER, PEM (X.509).
     - Загрузите файл приватного ключа. Допустимые форматы: DER, PEM (PKCS#8).
     - При необходимости укажите пароль для приватного ключа.
   - Контейнер PKCS#12. Загрузите файл сертификата. Допустимый формат: PKCS#12.
     - При необходимости укажите пароль для приватного ключа.
9. Нажмите на кнопку **Создать**. Переменная будет создана и добавлена в список в разделе **Ресурсы** → **Переменные**. Система отобразит соответствующее уведомление.

Для того чтобы переменную можно было добавлять в объекты системы, ее необходимо включить в карточке переменной.

Просмотр переменной

Чтобы просмотреть переменную:

1. Перейдите в раздел **Ресурсы** → **Переменные**. Система отобразит сведения об имеющихся переменных.
2. Нажмите на строку переменной в списке. Система отобразит страницу с карточкой этой переменной с подробной информацией о нем.

Название переменной отображается в верхней части карточки.

Чтобы закрыть карточку и вернуться к списку переменных, нажмите на крестик в правом верхнем

углу карточки.

Панель инструментов карточки включает в себя следующие компоненты:

- Выпадающее меню **Действия** (⋮) с опциями:
  - **Изменить** — изменение параметров переменной.
  - **Удалить** — удаление переменной.

Над списком полей находится переключатель, с помощью которого можно включить или выключить переменную. В объекты системы можно добавлять только включенные переменные.

Карточка содержит вкладку **Информация**, которая включает в себя следующие поля:

- **Название** — название переменной.
- **ID** — идентификатор переменной.
- **Статус** — статус переменной (Включен, Выключен).
- **Описание** — описание переменной.
- **Секрет** — индикатор, указывающий, является ли переменная секретом.
- **Тип** — тип переменной.
- **Дата создания** — дата и время создания переменной.
- **Создал** — пользователь, создавший переменную.
- **Дата изменения** — дата и время изменения переменной.
- **Изменил** — пользователь, изменивший переменную.

Панель **Значение** в нижней части карточки отображает значение переменной. Если переменная является секретом, панель не отображается.

Изменение переменной

Чтобы изменить переменную:

1. Перейдите в раздел **Ресурсы** → **Переменные**. Система отобразит сведения об имеющихся переменных.
2. Нажмите на строку переменной в списке. Система отобразит страницу с карточкой этой переменной с подробной информацией о ней.
3. Из выпадающего меню **Действия** (⋮) выберите опцию **Изменить**. Отобразится окно изменения переменной.
4. Внесите изменения в требуемые поля переменной.
5. Нажмите на кнопку **Сохранить**. Измененные данные переменной будут сохранены. Система отобразит соответствующее уведомление.

Изменить название и описание переменной можно и непосредственно из ее карточки:

1. Перейдите в раздел **Ресурсы** → **Переменные**. Система отобразит сведения об имеющихся переменных.

2. Нажмите на строку переменной в списке. Система отобразит в правой части экрана карточку этой переменной с подробной информацией о ней.
3. Нажмите на кнопку **Изменить** (✎) рядом с текущим названием или описанием переменной в его карточке. Система отобразит поле ввода значения.
4. Введите новое значение поля.
5. Нажмите **Сохранить**. Измененные данные переменной будут сохранены. Система отобразит соответствующее уведомление.

Удаление переменной

Чтобы удалить переменную:

1. Перейдите в раздел **Ресурсы** → **Переменные**. Система отобразит сведения об имеющихся переменных.
2. Нажмите на строку переменной в списке. Система отобразит страницу с карточкой этой переменной с подробной информацией.
3. Из выпадающего меню **Действия** (⋮) выберите опцию **Удалить**. Отобразится окно подтверждения удаления переменной.
4. Нажмите на кнопку **Удалить**. Переменная исчезнет из списка переменных. Система отобразит уведомление об удалении выбранной переменной.

Работа с конфигурациями

Данный раздел описывает процесс работы с конфигурациями ресурсов в системе. Работа осуществляется в разделе **Ресурсы** → **Конфигурации** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О конфигурациях

Конфигурация ресурсов — это компонент системы, позволяющий группировать аргументы в одну сущность. Конфигурация используется для задания структур и позволяет описывать параметры для установки соединения с внешними системами.

Конфигурации строятся на базе схем конфигураций и должны содержать значения для всех обязательных полей. Опциональные поля при этом могут быть пропущены. В качестве значений полей могут быть выбраны как значения, введенные пользователем при создании конфигурации, так и ссылки на переменные и секреты такого же типа, что и задаваемое значение

Просмотр переменных и секретов в конфигурациях

При использовании переменных и секретов в конфигурациях значения остаются скрытыми. Когда вы добавляете переменные и секреты в поля конфигурации, в полях отображаются только их имена. Сами значения будут подставляться только в момент запуска скрипта с этой конфигурацией. Таким образом, конфигурации удобно использовать одновременно при работе с большим количеством

скриптов. При изменении какого-либо поля конфигурации это нужно сделать только один раз.

При указании переменной или секрета в качестве значения поля конфигурации система отображает в выпадающем списке только переменные, соответствующие типу этого поля.

Скрипт может запускаться неоднократно — вручную или согласно параметрам триггера. Если в таком скрипте используется конфигурация с указанной в ней переменной или секретом, результаты работы скрипта могут отличаться от запуска к запуску. Это может произойти, если в промежутке между запусками значение переменной или секрета было изменено.

Интерфейс раздела

Панель инструментов включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет добавить конфигурацию в систему.
- Поле **Поиск** предназначено для быстрого поиска конфигураций в отображаемом списке по названию.
- Выпадающий список **Статус** позволяет отфильтровать конфигурации по статусу.

В рабочей области отображается таблица имеющихся конфигураций. Колонки таблицы представлены следующим образом:

- **Название** — название конфигурации.
- **Описание** — описание конфигурации.
- **Статус** — статус конфигурации.
- **Дата создания** — дата и время создания конфигурации.
- **Дата изменения** — дата и время изменения конфигурации.
- **Версия схемы** — версия схемы конфигурации.

При работе с таблицей конфигураций доступны следующие операции:

- Поиск конфигураций по полю **Название**.
- Фильтрация конфигураций по полю **Статус**. Статус выбирается из выпадающего списка.
- Сортировка конфигураций по полям **Название**, **Дата создания** и **Дата изменения**.

Работа с конфигурацией

Доступные операции над конфигурацией:

- создание конфигурации;
- изменение конфигурации;
- обновление конфигурации;
- удаление конфигурации.

Создание конфигурации

Чтобы создать конфигурацию:

1. Перейдите в раздел **Ресурсы** → **Конфигурации**.
2. Нажмите на кнопку **Создать** (+). Отобразится форма создания конфигурации.

   Для выбора доступны только схемы конфигурации со статусом **Включен**.

3. Добавьте в конфигурацию элемент экспертизы.

   Панель инструментов окна создания конфигурации включает в себя следующие компоненты:

   - Поле **Поиск** предназначено для быстрого поиска элементов экспертизы в отображаемом списке по названию и идентификатору.
   - Поле фильтра **Теги** позволяет задать комбинацию тегов искомых элементов.
   - Кнопка **Добавить** позволяет добавить выбранный элемент экспертизы в конфигурацию.

   Чтобы добавить элемент экспертизы:

   a. Раскройте дерево каталогов в левой части окна и выберите каталог, который содержит нужный элемент экспертизы.
   b. Выделите элемент экспертизы, который необходимо добавить в конфигурацию.
   c. Нажмите на кнопку **Добавить**. Выбранный элемент экспертизы будет добавлен в конфигурацию. Система отобразит окно параметров конфигурации.

4. Укажите параметры конфигурации.

   Панель инструментов в окне параметров конфигурации включает в себя следующие компоненты:

   - Кнопка **Добавить** (+) позволяет создать новую конфигурацию ресурса.
   - Кнопка **Отменить** позволяет отменить создание новой конфигурации.

     Чтобы указать параметры конфигурации:

     a. В блоке **Данные конфигурации** введите следующие данные:
        - Человекочитаемый идентификатор.
        - Название конфигурации.
        - Описание конфигурации.
     b. Введите значения полей в блоке **Настройка полей конфигурации**, учитывая типы полей. Поля конфигурации отображаются в этом блоке в соответствии с настройками схемы конфигурации.

5. Нажмите на кнопку **Добавить**. Конфигурация будет создана в системе. Система отобразит уведомление об успешном создании.

Просмотр конфигурации

Чтобы просмотреть конфигурацию:

1. Перейдите в раздел **Ресурсы** → **Конфигурации**. Система отобразит сведения об имеющихся конфигурациях.
2. Нажмите на строку конфигурации в списке. Система отобразит в правой части экрана карточку этой конфигурации с подробной информацией о ней.

Название конфигурации отображается в верхней части карточки.

Чтобы закрыть карточку и вернуться к списку конфигураций, нажмите на крестик в правом верхнем углу карточки.

Панель инструментов карточки включает в себя следующие компоненты:

- **Изменить** — изменение конфигурации.
- **Обновить** — обновление конфигурации.

  Обновить конфигурацию можно, если в системе опубликована более новая версия схемы конфигурации, на основе которой создана текущая конфигурация.

- **Удалить** — удаление конфигурации.

Над списком полей находится переключатель, с помощью которого можно включить или выключить конфигурацию. В объекты системы можно добавлять только включенные конфигурации.

Карточка содержит вкладку **Информация**, которая включает в себя следующие поля:

- **Название** — название конфигурации.
- **ID тенанта** — идентификатор тенанта, в котором был добавлена конфигурация.
- **ID** — идентификатор конфигурации.
- **Статус** — статус конфигурации (**Включен**, **Выключен**).
- **Версия схемы** — номер версии схемы конфигурации.
- **Описание** — описание конфигурации.
- **Дата создания** — дата и время создания конфигурации.
- **Создал** — пользователь, создавший конфигурацию.
- **Дата изменения** — дата и время изменения конфигурации.
- **Изменил** — пользователь, изменивший конфигурацию.

Обновление конфигурации

Вы можете обновить конфигурацию только в случае, если в системе была опубликована более новая версия схемы конфигурации, чем его текущая версия. Для обновления конфигурации нужно изменить версию используемой в ней схемы на более новую.

Система отслеживает публикацию новых версий схем конфигураций. После публикации новой версии схемы конфигурации система:

- отображает в карточке конфигурации информационное сообщение о выходе новой версии схемы конфигурации;
- делает активной команду **Обновить** в выпадающем меню **Действия** (⋮) в верхней части карточки конфигурации.

Чтобы обновить конфигурацию:

1. Перейдите в раздел **Ресурсы** → **Конфигурации**. Система отобразит сведения об имеющихся конфигурациях.
2. Нажмите на строку конфигурации в списке. Система отобразит в правой части экрана карточку этой конфигурации с подробной информацией о ней.
3. Выберите опцию **Обновить** в выпадающем меню **Действия** (⋮) в верхней части карточки конфигурации. Отобразится окно обновления конфигурации.

При обновлении конфигурации система сравнивает название и тип каждого аргумента в текущей и новой версиях:

- При совпадении данных система отображает ранее введенное значение или переменную.
- Если совпадение неполное или отсутствует, данные аргумента в полях конфигурации будут очищены.

4. Нажмите на кнопку **Обновить**. Система запустит переход на последнюю опубликованную версию схемы конфигурации и отобразит уведомление об успешном обновлении.

Изменение конфигурации

1. Перейдите в раздел **Ресурсы** → **Конфигурация**. Система отобразит сведения об имеющихся конфигурациях.
2. Нажмите на строку конфигурации в списке. Система отобразит страницу с карточкой этой конфигурации с подробной информацией о нем. Название конфигурации отображается в верхней части карточки.
3. Выберите опцию **Изменить** в выпадающем меню **Действия** (⋮) в верхней части карточки конфигурации. Отобразится окно настроек конфигурации.
4. Внесите изменения в требуемые поля конфигурации.
5. Нажмите на кнопку **Сохранить**. Измененные данные конфигурации будут сохранены. Система отобразит соответствующее уведомление.

Удаление конфигурации

1. Перейдите в раздел **Ресурсы** → **Конфигурация**. Система отобразит сведения об имеющихся конфигурациях.
2. Нажмите на строку конфигурации в списке. Система отобразит страницу с карточкой этой конфигурации с подробной информацией о ней. Название конфигурации отображается в верхней части карточки.
3. Выберите опцию **Удалить** в выпадающем меню **Действия** (⋮) в верхней части карточки

конфигурации. Отобразится окно подтверждения удаления конфигурации.

4. Нажмите на кнопку **Удалить**. Система отобразит уведомление об удалении выбранной конфигурации, и конфигурация исчезнет из списка конфигураций.

Работа с воркерами

Данный раздел описывает процесс работы с воркерами в системе. Работа осуществляется в разделе **Ресурсы** → **Воркеры** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О воркерах

Воркеры — это компоненты системы, выполняющие назначенные им задачи и использующие для этого выделенные ресурсы системы (определенное количество ядер CPU и объем памяти RAM). Такими задачами являются последовательности действий, заданные в скриптах и плейбуках системы. Ресурсы системы, выделенные для воркера, указываются в параметрах воркера.

После запуска скрипта и ввода аргументов система преобразует их в формат JSON, и передает скрипт или плейбук на выполнение в воркер. При запуске скрипта система назначает ему специальный тег воркера. Этот тип тегов отличается от типа тегов, используемых в разделе **Экспертиза** для группировки скриптов по бизнес-модели. Тег воркера, назначаемый скрипту системой, указывает, какой конкретный воркер будет выполнять этот скрипт.

Таким образом, с помощью менеджера воркеров можно назначать определенным воркерам скрипты для выполнения, например, по языкам, для выполнения которых они предназначены, или по компонентам системы.

- Если аргументы в скрипте отсутствуют, в воркер передается пустой JSON-файл.
- Если имеются непереопределенные аргументы по умолчанию, аргументы будут подставлены в JSON-файл.
- Если имеются переопределенные аргументы по умолчанию, именно они будут подставлены в JSON-файл.

Воркер обрабатывает скрипт или плейбук по следующей схеме:

1. После получения скрипта или плейбука воркер находит его исходный код в сервисе экспертизы по идентификатору, анализирует его и преобразует в готовый для выполнения код.
2. Если необходимо отдельно установить определенные зависимости скрипта, воркер производит установку зависимостей в рамках отдельной задачи. Для этого воркер скачивает зависимости из реестра зависимостей, установленного по умолчанию.

   Реестры зависимостей для скачивания могут храниться:

   - в сети Интернет;
   - в локальных репозиториях.

Установленные зависимости в дальнейшем хранятся на этом воркере. Они используются при последующих выполнениях скрипта на этом воркере, что существенно сокращает время работы.

Если скрипту назначить другой тег воркера, скрипт, возможно, будет передан на выполнение в другой воркер. Это повлечет за собой затрату дополнительного времени на установку зависимостей на этом воркере, так как ранее они не были на нем установлены.

Шаг 2 и все последующие шаги описывают работу воркера как со скриптом, так и с каждым модулем в составе плейбука.

3. После проверки наличия и установки зависимостей воркер проверяет, запускался ли этот скрипт ранее с такими же параметрами и сохранились ли результаты этого запуска в кэше. Кэшированные данные хранятся в системе 7 дней.

   Если такие данные сохранены в кэше, воркер предоставит их, что ускорит выполнение процесса.

4. Если готовый результат не найден, задача передается на выполнение. Каждый воркер в отдельный момент времени выполняет только одну задачу.
5. Воркер запускает процесс с интерпретатором требуемого языка. В файловой системе в отдельной директории сохраняются все необходимые данные. Например, аргументы, ссылки на зависимости, необходимые для запуска и указанные пользователем переменные окружения.
6. Система помещает скрипт в специальную оболочку, позволяющую через библиотеку playbooks_sdk бесшовно получать доступ к переменным, элементам экспертизы, запускам и интеграциям с другими сервисами.
7. Скрипт выполняется. Становятся доступными результат выполнения скрипта и логи. Изменяется статус запуска.
8. Воркер завершает процесс, кэширует результаты и удаляет директорию с артефактами работы, так как все необходимые данные уже сохранены. После этого воркер переходит в режим ожидания следующей задачи.

- Если контейнер воркера пересоздается, все установленные зависимости теряются.
- Если контейнер воркера останавливается и перезапускается, все установленные зависимости сохраняются.

Интерфейс раздела

Панель инструментов включает в себя следующие компоненты:

- Кнопка **Добавить** (+) позволяет запустить выполнение воркера.
- Кнопка **Обновить** ( ) позволяет обновить воркеры до последней версии. Если все воркеры обновлены до последней версии, кнопка неактивна.
- Поле **Поиск** предназначено для быстрого поиска воркеров в отображаемом списке по названию.
- Выпадающий список **Статус** позволяет отфильтровать воркеры по статусу.

В рабочей области отображается таблица имеющихся воркеров. Колонки таблицы представлены

следующим образом:

- **Название** — название воркера.
- **Версия** — версия воркера. Если доступно обновление воркера до последней версии, рядом с версией отображается соответствующее уведомление.
- **Группа воркеров** — группа, в состав которой входит воркер.
- **Старт** — дата и время старта выполнения конкретной задачи воркером.
- **Выполнено задач** — количество задач, обработанных воркером в течение одного календарного дня.
- **Дата создания** — дата и время создания воркера.
- **Дата изменения** — дата и время изменения воркера.
- **Состояние** — состояние воркера.
- **Статус** — статус воркера (**Включен**, **Выключен**).
- **Описание** — описание воркера.

При работе с таблицей воркеров доступны следующие операции:

- Поиск воркеров по полю **Название**.
- Фильтрация воркеров по полям **Статус** и **Состояние**. Значения выбираются из выпадающего списка.
- Сортировка воркеров по полям **Название**, **Версия**, **Дата создания**, **Дата изменения**, **Состояние** и **Статус**.

При выборе конкретного воркера на экране открывается карточка с детальной информацией о нем.

Работа с воркером

Доступные операции над воркером:

- Создание воркера
- Просмотр воркера
- Обновление воркера
- Изменение воркера
- Удаление воркера

Создание воркера

Чтобы создать воркер:

1. Перейдите в раздел **Ресурсы** → **Воркеры**.
2. Нажмите на кнопку **Добавить** (＋). Отобразится форма создания воркера.
3. Введите название воркера.
4. Введите количество ядер процессора, выделенное на обработку скрипта. Количество ядер не может превышать 48.

5. Введите объем памяти RAM в МБ, выделенный на обработку скрипта. Объем памяти должен быть в диапазоне от 256 МБ до 524288 МБ.
6. Введите описание воркера.
7. Введите группу воркеров. По умолчанию поле заполняется значением **all**.
8. Переведите переключатель **Дополнительные репозитории** в активное положение, чтобы указать пользовательские репозитории, из которых вы хотите импортировать пакеты языка Python или языка TypeScript. После того как пакеты будут импортированы из указанных репозиториев, их можно использовать для создания скриптов и плейбуков на этих языках.

   Когда переключатель находится в активном положении, система отображает поля, в которых нужно ввести данные соответствующих репозиториев:

   - **Python**;
   - **TypeScript (Bun)**;
   - **TypeScript (Deno)**.
9. Переведите переключатель **Настройки прокси** в активное положение, чтобы задать переменные для серверов прокси. Указанные переменные будут использоваться на этом воркере.

   Когда переключатель находится в активном положении, система отображает поля, в которых нужно ввести переменные для серверов прокси:

   - **Http proxy** — задается переменная `http_proxy`, содержащая URL сервера прокси для протокола HTTP.
   - **Https proxy** — задается переменная `https_proxy`, содержащая URL сервера прокси для протокола HTTPS.
   - **No proxy** — задается переменная `no_proxy`, содержащая разделенный запятыми список доменов, для которых сервер прокси не должен использоваться.
10. Переведите переключатель **Переменные окружения** в активное положение, чтобы задать произвольные переменные окружения. Система отобразит таблицу для ввода пар ключей и значений переменных окружения.

    Для задания переменной окружения:

    a. Нажмите на кнопку ＋.
    b. Заполните поля **Ключ** и **Значение** данными новой переменной окружения.
    c. Нажмите на кнопку **Сохранить**. Введенная переменная будет использоваться на этом воркере.
    d. Выполните шаги a—c для добавления других переменных окружения.
11. Нажмите на кнопку **Создать**. Воркер будет добавлен в систему. Система отобразит соответствующее уведомление.

Просмотр воркера

Чтобы просмотреть воркер:

1. Перейдите в раздел **Ресурсы** → **Воркеры**. Система отобразит сведения об имеющихся воркерах.

2. Нажмите на строку воркера в списке. Система отобразит справа карточку этого воркера с подробной информацией о нем.

Название воркера отображается в верхней части карточки.

Чтобы закрыть карточку и вернуться к списку воркеров, нажмите на крестик в правом верхнем углу карточки.

В правом верхнем углу карточки расположена кнопка действий (⋮). При нажатии на нее открывается выпадающее меню со следующими действиями:

- Кнопка **Изменить** ( ) позволяет изменить воркер.
- Кнопка **Обновить** ( ) позволяет обновить воркер. Кнопка доступна, если версия воркера отличается от последней версии и воркер включен.
- Кнопка **Удалить** ( ) позволяет удалить воркер.

Карточка воркера содержит следующие вкладки:

- **Информация** — основная информация о воркере.
- **Окружение** — сведения о пользовательских репозиториях и переменных, заданных при создании воркера.

Вкладка Информация

На вкладке отображается характеристика процессора (CPU), объем оперативной памяти (RAM) и жесткого диска (HDD), а также их уровень загруженности.

Над списком полей находится переключатель, с помощью которого можно включить или выключить воркер.

- Если воркер участвует в процессе обработки задачи, выключить его нельзя.
- Если воркер является последним в группе воркеров, на которой запускаются задания по триггеру, при его выключении задания не будут выполнены.

Если версия воркера отличается от последней версии, на вкладке отображается соответствующее предупреждение.

Также вкладка включает в себя следующие поля:

- **ID тенанта** — идентификатор тенанта, в котором был добавлен воркер.
- **Название** — название воркера.
- **Описание** — описание воркера.
- **Версия** — версия воркера.
- **Состояние** — состояние воркера.
- **Группа воркеров** — группа, в состав которой входит воркер.
- **Старт** — дата и время старта выполнения конкретной задачи воркером.
- **Выполнено задач** — количество задач, обработанных воркером в течение одного календарного

дня.

- **Дата создания** — дата и время создания воркера.
- **Создал** — пользователь, создавший воркер.
- **Дата изменения** — дата и время изменения воркера.
- **Изменил** — пользователь, изменивший воркер.

Вкладка Окружение

На вкладке отображаются сведения о пользовательских репозиториях и переменных, заданных при создании воркера.

- В разделе **Дополнительные репозитории** отображаются пользовательские репозитории, из которых импортируются пакеты на определенных языках. Эти пакеты можно использовать для создания скриптов и плейбуков на указанных языках.
- В разделе **Настройки прокси** отображаются переменные, содержащие URL серверов прокси для различных протоколов и список доменов, для которых сервер прокси не должен использоваться.
- В разделе **Переменные окружения** отображаются переменные окружения.

Обновление воркера

Вы можете обновить воркеры, имеющиеся в системе, до их последних версий. Процесс обновления воркеров можно запустить:

- со страницы **Воркеры**;
- из карточки воркера.

Обновление воркеров со страницы Воркеры

Чтобы обновить воркеры со страницы **Воркеры**:

1. Перейдите в раздел **Ресурсы** → **Воркеры**. Система отобразит сведения об имеющихся воркерах.
2. Если необходимо обновить только определенные воркеры, выберите их в списке.
3. Нажмите на кнопку **Обновить** (🔄) над списком воркеров. Система отобразит меню обновления со следующими командами:
   - **Обновить все** — позволяет обновить все воркеры неактуальных версий. Рядом с командой отображается количество воркеров в системе, не обновленных до актуальной версии.
   - **Обновить выбранные** — позволяет обновить воркеры, предварительно выбранные в списке на шаге 2. Если не выбран ни один воркер, кнопка недоступна.
4. Выберите команду обновления из меню. Система запустит процесс обновления.

Воркеры в системе могут находиться в различных состояниях, от которых зависят доступность и работоспособность воркеров. Состояние воркера отображается в таблице воркеров в колонке **Состояние**. Ознакомьтесь с таблицей состояний воркеров.

Обновление воркера из его карточки

Чтобы обновить воркер из его карточки:

1. Перейдите в раздел **Ресурсы** → **Воркеры**. Система отобразит сведения об имеющихся воркерах.
2. Нажмите на строку воркера в списке. Система отобразит справа карточку этого воркера с подробной информацией о нем. Название воркера отображается в верхней части карточки.
3. Нажмите на кнопку **Действия** (⋮). Система отобразит меню команд.
4. Выберите команду **Обновить воркер**. Система запустит процесс обновления.

Процесс обновления воркеров

Этап подготовки к обновлению

После запуска команды обновления система переводит воркеры в состояние **Профилактика**, отображает окно подготовки к обновлению воркеров и проверяет воркеры на наличие активных заданий. Воркер в состоянии **Профилактика** недоступен для выполнения заданий на нем. После окончания проверки система закроет окно и отобразит окно обновления воркеров.

Кнопка **Закрыть** позволяет закрыть окно подготовки к обновлению воркеров, не дожидаясь его автоматического закрытия.

После окончания проверки система отобразит окно обновления воркеров. Оно содержит таблицу со сведениями о количестве воркеров в различных статусах обновления.

Поле **Поиск** предназначено для быстрого поиска воркеров в отображаемом списке по названию.

Колонки таблицы представлены следующим образом:

- **Название** — название воркера.
- **Старая версия** — текущая версия воркера.
- **Новая версия** — актуальная версия, до которой можно обновить воркер.
- **Статус обновления** — статус обновления воркера.

Если необходимо отменить обновление какого-либо воркера, наведите курсор на строку этого воркера в таблице и нажмите на отобразившуюся кнопку **Отменить обновление**. После подтверждения отмены система удалит воркер из таблицы.

Сформировав список воркеров, который требуется обновить, нажмите на кнопку **Обновить**. Обновление воркеров будет запущено. Система отобразит в окне информацию о количестве воркеров в различных статусах обновления.

Отменить обновление воркеров можно, нажав на кнопку **Отмена** или крестик в правом верхнем углу окна.

Отменить обновление после этапа подготовки нельзя.

Этап обновления

После нажатия кнопки **Обновить** система начнет обновление воркеров, которые можно обновить

немедленно. Такие воркеры переходят в статус обновления **Обновление**, а затем — в **Обновлен**.

Пользователь может выполнять определенные действия над воркерами, обновление которых не начато. Кнопки, запускающие эти действия, отображаются при наведении курсора на строку этого воркера.

- Если воркер в таблице имеет статус обновления **Завершается задание**, для воркера доступны следующие кнопки:
  - **Завершить задание** — позволяет остановить выполнение задания с последующим обновлением воркера. Задание будет завершено с ошибкой.
  - **Отменить обновление** — позволяет отменить обновление воркера.

Вы можете не выполнять над воркером в этом статусе никаких действий. Система начнет обновление воркера после завершения выполнения задания.

- Если задание на воркере выполнено с ошибкой или воркер не может завершить выполнение задания, система устанавливает для воркера статус обновления **Ошибка завершения задания**. Необходимо принудительно завершить выполнение задания на воркере или отменить обновление воркера.

  Для воркера доступны следующие кнопки:

  - **Принудительно завершить** — позволяет принудительно завершить выполнение задания на воркере.

При принудительном завершении система удалит все связи с воркером и пересоздаст воркер с обновлением до актуальной версии.

  - **Отменить обновление** — позволяет отменить обновление воркера.

Система обновит воркеры со статусами обновления, для которых это возможно.

Таблица 11. Таблица статусов обновления воркеров на различных этапах обновления

Этапы обновления	Статус обновления	Описание статуса	Доступные действия
Подготовка к обновлению	Готов к обновлению	У воркера нет активных заданий. Он готов к обновлению	• Отменить обновление
	Найдены активные задания	У воркера найдены активные задания	• Отменить обновление

Этапы обновления	Статус обновления	Описание статуса	Доступные действия
Обновление	Завершается задание	Воркер находится в процессе завершения задания. После завершения задания он будет обновлен	• Отменить обновление • Завершить выполнение
	Обновление	У воркера нет активных заданий. Он находится в процессе обновления	• Отменить обновление
	Ошибка завершения задания	Воркер не смог завершить задание. Обнаружены ошибки	• Отменить обновление • Завершить принудительно
Результат обновления	Ошибка обновления	Воркер обновить не удалось	• Нет
	Обновлен	Воркер успешно обновлен	• Нет

Таблица 12. Таблица состояний воркеров

Состояние	Индикатор	Включен	Выключен	Переключение между статусами	Описание состояния
Активен	Зеленый	+	—	Доступно	Работоспособное состояние воркера: • готов к выполнению задач; • выполняет задачи; • готов к профилактическим работам. Если при выключении воркера выполнялись задачи, они переводятся в статус ошибки.

Состояние	Индикатор	Включен	Выключен	Переключение между статусами	Описание состояния
Недоступен	Серый	—	+	Доступно	• Воркер не существует. Он недоступен для выполнения функций и профилактических работ. • При каждом выключении образ воркера удаляется, а настройки сохраняются.
Профилактика	Синий	+	—	Недоступно	• Выполняются следующие операции над воркером: ◦ Создание воркера. При включении создается новый воркер, настройки сохраняются с предыдущего воркера. ◦ Обновление воркера. ◦ Принудительная остановка воркера. • Работоспособность: ◦ Воркер не готов выполнять новые задачи. ◦ Воркер не может завершать выполнение поставленных задач.
Ошибка	Красный	+	—	Доступно	• Воркер не создался и сервисы для воркера недоступны. Ошибка не связана с обновлением воркера. • Работоспособность: воркер не готов выполнять новые задачи.

Состояние	Индикатор	Включен	Выключен	Переключение между статусами	Описание состояния
Ошибка обновления	Красный	+	—	Доступно	• Возникла ошибка при обновлении воркера. • Работоспособность: ◦ Воркер необновленной версии готов выполнять новые задачи. ◦ Когда воркер берет задачи в работу, он переходит в статус **Активен** с необновленной версией.

Изменение воркера

Чтобы изменить воркер:

1. Перейдите в раздел **Ресурсы** → **Воркеры**. Система отобразит сведения об имеющихся воркерах.
2. Нажмите на строку воркера в списке. Система отобразит справа карточку этого воркера с подробной информацией о нем. Название воркера отображается в верхней части карточки.
3. Нажмите на кнопку **Действия** (⋮). Система отобразит меню команд.
4. Выберите команду **Изменить** (). Отобразится окно изменения воркера.
5. Внесите изменения в требуемые поля воркера.
6. Нажмите на кнопку **Сохранить**. Измененные данные воркера будут сохранены. Система отобразит соответствующее уведомление.

Удаление воркера

Чтобы удалить воркер:

1. Перейдите в раздел **Ресурсы** → **Воркеры**. Система отобразит сведения об имеющихся воркерах.
2. Нажмите на строку воркера в списке. Система отобразит справа карточку этого воркера с подробной информацией о нем. Название воркера отображается в верхней части карточки.
3. Нажмите на кнопку **Действия** (⋮). Система отобразит меню команд.
4. Выберите команду **Удалить** (). Отобразится окно подтверждения удаления воркера.
5. Нажмите на кнопку **Удалить**. Выбранный воркер исчезнет из списка воркеров. Система отобразит уведомление об удалении воркера.

Работа со сканерами

Управление сканерами осуществляется в разделе **Ресурсы** → **Сканеры**.

О сканерах

Сканер — это компонент системы, который собирает сведения об оборудовании во внешних системах и передает их в систему для обработки.

Работа с разделом

Панель инструментов включает в себя следующие компоненты:

- Кнопка **Создать** позволяет создать новый сканер.
- Поле **Поиск** предназначено для быстрого поиска сканеров в отображаемом списке по названию.

В рабочей области отображается таблица имеющихся сканеров, которая содержит следующие столбцы:

- **Название** сканера.
- **Пространство** сканера.
- **Активен** — сведения о статусе активности сканера.
- **Описание** сканера.

Работа со списком сканеров

Доступны следующие действия со сканерами:

- поиск сканеров;
- создание сканера;
- просмотр сканера;
- изменение сканера;
- удаление сканера.

Поиск сканеров

Чтобы найти сканер, введите его название в строку поиска над списком сканеров. В списке отобразятся сканеры, в названии которых встречается введенный запрос.

Создание сканера

Чтобы создать сканер:

1. Нажмите на кнопку **Создать** (┼). Отобразится форма создания сканера.
2. Укажите название сканера.
3. При необходимости укажите описание сканера.

4. Из раскрывающегося списка выберите пространство, в котором будет работать сканер.
5. Нажмите на кнопку **Сохранить**. Созданный сканер отобразится в списке сканеров и станет доступен для использования.

Просмотр сканера

Чтобы просмотреть сканер, нажмите на строку сканера в списке. Система отобразит в правой части экрана карточку сканера с подробной информацией о нем.

Название сканера отображается в верхней части карточки.

Панель инструментов включает в себя следующие компоненты:

- **Удалить** ( ) — удаление сканера.

Карточка содержит разделы:

- **Информация** — сведения о характеристиках сканера и пользователях, создавших/изменивших сканер.
- **Сервисы** — сведения о сервисах, с которыми работает сканер.

Изменение сканера

Чтобы просмотреть сканер, нажмите на строку сканера в списке. Система отобразит в правой части экрана карточку сканера с подробной информацией о нем.

В разделе **Сервисы** вкладки **Информация** отобразятся сервисы, с которыми работает сканер.

Чтобы удалить сервис, с которым работает сканер:

1. Наведите курсор на строку сервиса. Справа отобразится кнопка удаления сервиса ( ).
2. Нажмите на кнопку. Сервис будет удален.

Удаление сканера

1. Чтобы удалить сканер, нажмите на строку сканера в списке. Система отобразит карточку сканера с подробной информацией о нем.
2. Нажмите на кнопку **Удалить** ( ) на панели инструментов карточки. Отобразится окно подтверждения удаления сканера.
3. Нажмите на кнопку **Удалить**. Система отобразит уведомление об удалении выбранного сканера, и он исчезнет из списка сканеров.

Агенты

В данном разделе описана работа с менеджером агентов в системе. Менеджер позволяет управлять агентами и их группами, а также настраивать шлюзы для подключения агентов к кластерам.

Содержание раздела:

- Работа с агентами
- Работа с группами агентов
  - Настройка конфигурации группы агентов
  - Синтаксис языка XPath
- Работа со шлюзами

Работа с агентами

Данный раздел описывает процесс работы с агентами в системе. Работа осуществляется в разделе **Агенты** → **Агенты** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Об агентах

Агенты устанавливаются на конечные устройства и используются для выявления угроз и осуществления реагирования на эти угрозы. Они позволяют осуществлять сетевую изоляцию устройства и производить сбор информации об уязвимостях и угрозах.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Обновить данные** позволяет обновить список агентов.
- Поле **Поиск** предназначено для быстрого поиска агентов в отображаемом списке по имени, группе, IP-адресу или ОС.
- Кнопка **Фильтр** ( ) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка агентов.
- Кнопки команд над выбранными в таблице агентами (становятся доступными после выбора хотя бы одного агента из списка):
  - — удаляет выбранные агенты;
  - — создает группу с выбранными агентами;
  - — добавляет выбранные агенты в группу;
  - — удаляет выбранные агенты из группы.

В рабочей области раздела отображается таблица имеющихся агентов. Таблица содержит следующие столбцы:

- **Название** — название агента, используемое для его идентификации. Слева от названия отображается цветовой индикатор статуса агента:
  - **Выключен** — красный;
  - **Офлайн** — серый;
  - **Онлайн** — зеленый;
  - **Зарегистрирован** — желтый.
- **Группа** — список групп, в которые включен агент.
- **IP-адрес** — список IP-адресов узлов на агенте.
- **ОС** — операционная система, под которой работает агент.
- **Архитектура** — архитектура процессора агента.
- **ID машины** — уникальный идентификационный код компьютера агента.
- **ID агента** — уникальный идентификационный код агента.
- **Версия агента** — версия агента.
- **Дата регистрации** — дата и время регистрации агента.
- **Последняя активность** — дата и время последней активности агента.
- **Тенант** — название тенанта, к которому относится агент.

При работе с таблицей агентов доступны следующие операции:

- Поиск агентов по полям **Название**, **Группа**, **IP-адрес**, **ОС**.
- Фильтрация агентов по полям **Статус**, **Группа**, **Тип ОС**, **Версия агента**, **Дата регистрации**, **Последняя активность**, **Тенант**.
  - Статус, группа, тип ОС, версия агента и последняя активность выбираются из выпадающих списков.
  - Начало и конец диапазона дат регистрации задаются с помощью панели выбора даты и времени.
- Сортировка агентов по полям **Название**, **Группа**, **IP-адрес**, **ОС**, **Архитектура**, **Версия агента**, **Дата регистрации**, **Последняя активность**.
- Настройка отображения таблицы агентов.

При выборе конкретного агента в правой части рабочей области отображается его карточка с детальной информацией.

В верхнем правом углу карточки расположена кнопка действий (⋮). При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранного агента.

Карточка агента содержит следующие вкладки:

- **Общее** — основная информация об агенте.
- **Процессы** — список активных процессов на узле.

- **Порты** — список портов на узле.
- **Контейнеры** — список Docker-контейнеров на агенте.

Работа с агентом

Доступные операции над агентами:

- Просмотр агента
- Обновление данных об агенте
- Удаление агента
- Создание группы на базе агента
- Добавление агента в группу
- Удаление агента из группы

Просмотр агента

Чтобы просмотреть агент:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.

Название агента отображается в верхней части карточки.

Карточка агента содержит следующие вкладки:

- **Общее** — основная информация об агенте.
- **Процессы** — список активных процессов на узле.
- **Порты** — список портов на узле.
- **Контейнеры** — список Docker-контейнеров на агенте.

Вкладка Общее

На вкладке **Общее** отображается основная информация об агенте. Вкладка содержит сведения, сгруппированные по следующим секциям:

- Метрики агента:
  - **CPU** — количество ядер процессора агента.
  - **RAM** — объем оперативной памяти агента.
  - **HDD** — объем жесткого диска агента.
- Основная информация об агенте:
  - **ID агента** — уникальный идентификационный код агента.
  - **Название** — имя агента, используемое для его идентификации в системе.

- **ОС** — операционная система, под которой работает агент.
- **Архитектура** — архитектура процессора агента.
- **Версия агента** — версия агента.
- **Статус** — текущий статус агента (выключен, офлайн, онлайн или зарегистрирован).
- **IP-адрес** — список IP-адресов узлов на агенте.
- **Группа** — список групп, в которые включен агент.
- **Тенант** — название тенанта, к которому относится агент.
- **ID машины** — уникальный идентификационный код компьютера агента.
- **Дата регистрации** — дата и время регистрации агента.
- **Последняя активность** — дата и время последней активности агента.

- Список пользователей на агенте. Список представляет собой таблицу со следующими столбцами:

  - **Логин** — логин пользователя.
  - **Последняя активность** — дата и время последнего входа пользователя в систему.

  Для быстрого поиска пользователей в списке введите запрос в строку поиска, расположенную над таблицей. В списке отобразятся пользователи, в логинах или ФИО которых с учетом регистра встречается введенный запрос.

В нижней части вкладки расположена кнопка **Подробнее**, при нажатии на которую открывается более подробная карточка агента.

Подробную карточку также можно открыть, дважды нажав на строку агента в таблице раздела **Агенты → Агенты**.

Панель инструментов подробной карточки содержит кнопки команд над агентом:

- — удаляет агент;
- — обновляет инвентарную информацию об агенте;
- — создает группу с агентом;
- — добавляет агент в группу;
- — удаляет агент из группы.

При открытии вкладки **Общее** в подробной карточке агента таблица пользователей на агенте будет содержать следующие колонки:

- **Логин** — логин пользователя.
- **Ф.И.О** — фамилия, имя и отчество пользователя.
- **Последняя активность** — дата и время последнего входа пользователя в систему.

Вкладка Процессы

На вкладке **Процессы** отображается список активных процессов на узле. Список представляет

собой таблицу со следующими столбцами:

- **PID** — уникальный идентификатор процесса.
- **Название** — название процесса.

При открытии вкладки **Процессы** в подробной карточке агента таблица процессов будет содержать следующие столбцы:

- **PID** — уникальный идентификатор процесса.
- **Название** — название процесса.
- **Путь** — подробный путь процесса.

Для быстрого поиска процессов в списке введите запрос в строку поиска, расположенную над таблицей. В списке отобразятся процессы, в полях которых встречается введенный запрос.

Вкладка Порты

На вкладке **Порты** отображается список портов на узле. Список представляет собой таблицу со следующими столбцами:

- **Адрес** — адрес порта.
- **Сервис** — имя сервиса, работающего на порте. При наведении курсора на имя сервиса отображается всплывающее сообщение с его подробной информацией.
- **Протокол** — сетевой протокол, используемый на порте.

При открытии вкладки **Порты** в подробной карточке агента таблица портов будет содержать следующие столбцы:

- **Адрес** — адрес порта.
- **Имя порта** — имя порта.
- **Порт** — порт подключения.
- **Протокол** — сетевой протокол, используемый на порте.

Для быстрого поиска портов в списке введите запрос в строку поиска, расположенную над таблицей. В списке отобразятся порты, в полях которых встречается введенный запрос.

Вкладка Контейнеры

На вкладке **Контейнеры** отображается список Docker-контейнеров на агенте. Список представляет собой таблицу со следующими столбцами:

- **Название** — название контейнера. Слева от названия отображается цветовой индикатор статуса контейнера:
  - **Работает** — зеленый;
  - **Не работает** — серый.
- **Образ** — имя образа контейнера. При наведении курсора на строку с именем образа отображается всплывающее сообщение с его подробной информацией.

Если для контейнера указаны порты, то в правой части его строки отобразится иконка ⇠⇢. При наведении курсора на эту иконка отображается всплывающее сообщение с информацией о портах.

Пример 40. Формат вывода информации о порте

```
<IP-адрес сервера>:<порт сервера> -> <порт контейнера>/<протокол>
```

IP-адрес и порт сервера отображаются только в том случае, если порт контейнера транслируется на сервер.

При открытии вкладки **Контейнеры** в подробной карточке агента таблица контейнеров будет содержать следующие столбцы:

- **Название** — название контейнера.
- **Статус** — текущий статус контейнера: **Работает** или **Не работает**.
- **Команда** — команда, выполняемая при запуске контейнера.
- **Образ** — имя образа контейнера. При наведении курсора на строку с именем образа отображается всплывающее сообщение с его подробной информацией.
- **Порты** — порты, указанные для контейнера.

  Пример 41. Формат вывода информации о порте

  ```
  <IP-адрес сервера>:<порт сервера> -> <порт контейнера>/<протокол>
  ```

  IP-адрес и порт сервера отображаются только в том случае, если порт контейнера транслируется на сервер.

Для быстрого поиска контейнеров в списке введите запрос в строку поиска, расположенную над таблицей. В списке отобразятся контейнеры, в полях которых встречается введенный запрос.

Обновление данных об агенте

Чтобы обновить инвентарную информацию об агенте:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Обновить инвентарку**. Система отправит запрос на обновление инвентарной информации о выбранном агенте и отобразит соответствующее уведомление.

Обновить инвентарную информацию об агенте можно также из его подробной карточки. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.

2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. На экране отобразятся подробные сведения об агенте.
4. Нажмите на кнопку на панели инструментов. Система отправит запрос на обновление инвентарной информации о выбранном агенте и отобразит соответствующее уведомление.

Удаление агента

Чтобы удалить агент:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления агента.
4. Нажмите на кнопку **Удалить**. Система удалит агент и отобразит соответствующее уведомление. Удаленный агент будет исключен из списка раздела.

Удалить агент можно также из его подробной карточки. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. На экране отобразятся подробные сведения об агенте.
4. Нажмите на кнопку на панели инструментов. Отобразится окно подтверждения удаления агента.
5. Нажмите на кнопку **Удалить**. Система удалит агент и отобразит соответствующее уведомление. Удаленный агент будет исключен из списка раздела.

Вы также можете удалить сразу несколько агентов. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. В левом столбце таблицы установите флажки напротив тех агентов, которые вы хотите удалить.
3. Нажмите на кнопку на панели инструментов. Отобразится окно подтверждения удаления агентов.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные агенты и отобразит соответствующее уведомление. Удаленные агенты будут исключены из списка раздела.

Создание группы на базе агента

В разделе **Агенты** → **Агенты** доступно создание группы на базе агента.

Чтобы создать группу на базе агента:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Создать группу**. Отобразится окно создания группы.
4. Задайте название группы.
5. Если после создания группы вы хотите перейти к настройке ее конфигурации, позволяющей отслеживать различные действия на агентах, установите флажок **Перейти к конфигурации**.
6. Нажмите на кнопку **Создать**. Система создаст группу с выбранным агентом и отобразит соответствующее уведомление. Созданная группа отобразится в разделе **Агенты** → **Группы агентов**.

Создать группу на базе агента можно также из его подробной карточки. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. На экране отобразятся подробные сведения об агенте.
4. Нажмите на кнопку ＋ на панели инструментов. Отобразится окно создания группы.
5. Задайте название группы.
6. Если после создания группы вы хотите перейти к настройке ее конфигурации, позволяющей отслеживать различные действия на агентах, установите флажок **Перейти к конфигурации**.
7. Нажмите на кнопку **Создать**. Система создаст группу с выбранным агентом и отобразит соответствующее уведомление. Созданная группа отобразится в разделе **Агенты** → **Группы агентов**.

Вы также можете создать группу сразу из нескольких агентов. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. В левом столбце таблицы установите флажки напротив тех агентов, из которых вы хотите сформировать группу.
3. Нажмите на кнопку ＋ на панели инструментов. Отобразится окно создания группы.
4. Задайте название группы.

5. Если после создания группы вы хотите перейти к настройке ее конфигурации, позволяющей отслеживать различные действия на агентах, установите флажок **Перейти к конфигурации**.
6. Нажмите на кнопку **Создать**. Система создаст группу с выбранным агентом и отобразит соответствующее уведомление. Созданная группа отобразится в разделе **Агенты → Группы агентов**.

Добавление агента в группу

Чтобы добавить агент в группу:

1. Перейдите в раздел **Агенты → Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Добавить в группу**. Отобразится окно добавления агента в группу.
4. Выберите из выпадающего списка группу, в которую необходимо добавить агент.
5. Нажмите на кнопку **Добавить**. Система добавит агент в выбранную группу и отобразит соответствующее уведомление.

Добавить агент в группу можно также из его подробной карточки. Для этого:

1. Перейдите в раздел **Агенты → Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. На экране отобразятся подробные сведения об агенте.
4. Нажмите на кнопку на панели инструментов. Отобразится окно добавления агента в группу.
5. Выберите из выпадающего списка группу, в которую необходимо добавить агент.
6. Нажмите на кнопку **Добавить**. Система добавит агент в выбранную группу и отобразит соответствующее уведомление.

Вы также можете добавить в группу сразу несколько агентов. Для этого:

1. Перейдите в раздел **Агенты → Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. В левом столбце таблицы установите флажки напротив тех агентов, которые вы хотите добавить в группу.
3. Нажмите на кнопку на панели инструментов. Отобразится окно добавления агентов в группу.
4. Выберите из выпадающего списка группу, в которую необходимо добавить агенты.
5. Нажмите на кнопку **Добавить**. Система добавит выделенные агенты в выбранную группу и

отобразит соответствующее уведомление.

Удаление агента из группы

Чтобы удалить агент из группы:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить из группы**. Отобразится окно удаления агента из группы.
4. Выберите группу из выпадающего списка, из которой необходимо удалить агент.
5. Нажмите на кнопку **Удалить**. Система удалит агент из выбранной группы и отобразит соответствующее уведомление.

Удалить агент из группы можно также из его подробной карточки. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. Нажмите на строку агента в списке. Система отобразит в правой части экрана карточку этого агента с подробной информацией о нем.
3. Нажмите на кнопку **Подробнее** в нижней части карточки. На экране отобразятся подробные сведения об агенте.
4. Нажмите на кнопку на панели инструментов. Отобразится окно удаления агента из группы.
5. Выберите группу из выпадающего списка, из которой необходимо удалить агент.
6. Нажмите на кнопку **Удалить**. Система удалит агент из выбранной группы и отобразит соответствующее уведомление.

Вы также можете удалить из группы сразу несколько агентов. Для этого:

1. Перейдите в раздел **Агенты** → **Агенты**. Система отобразит сведения об имеющихся агентах, в том числе их текущий статус.
2. В левом столбце таблицы установите флажки напротив тех агентов, которые вы хотите удалить из группы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно удаления агентов из группы.
4. Выберите группу из выпадающего списка, из которой необходимо удалить агенты.
5. Нажмите на кнопку **Удалить**. Система удалит выделенные агенты из выбранной группы и отобразит соответствующее уведомление.

Работа с группами агентов

Данный раздел описывает процесс работы с группами агентов в системе. Работа осуществляется в разделе **Агенты** → **Группы агентов** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О группах агентов

Группа агентов объединяют агенты для упрощения выполнения массовых операций над ними.

Для групп агентов доступна настройка конфигурации, которая позволяет задавать политики отслеживания различных действий на агентах в рамках группы.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новую группу агентов.
- Поле **Поиск** предназначено для быстрого поиска групп агентов в отображаемом списке по названию и количеству агентов в группе.
- Кнопка **Фильтр** ( ) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка групп.
- Кнопка позволяет удалить выбранные группы агентов. Становится доступной после выбора хотя бы одной группы из списка.

В рабочей области раздела отображается таблица имеющихся групп агентов. Таблица содержит следующие столбцы:

- **Название** — название группы агентов, используемое для ее идентификации
- **Количество агентов** — количество агентов, входящих в группу.
- **Автоназначение** — индикатор включения автоматического назначения группы агенту (включено или выключено).
- **Дата создания** — дата и время создания группы.
- **Дата изменения** — дата и время последнего изменения группы.
- **Тенант** — название тенанта, к которому относится группа.

При работе с таблицей групп агентов доступны следующие операции:

- Поиск групп по полям **Название** и **Количество агентов**.
- Фильтрация групп по полям **Количество агентов**, **Автоназначение**, **Дата создания**, **Дата изменения**, **Тенант**.
  - Начало и конец диапазона количества агентов задаются вручную или с помощью кнопок — и + .

    - Поведение автоназначения выбирается из выпадающего списка.
    - Начало и конец диапазона дат создания и изменения задаются с помощью панели выбора даты и времени.
- Сортировка агентов по полям **Название**, **Количество агентов**, **Дата создания**, **Дата изменения**.
- Настройка отображения таблицы групп.

При выборе конкретной группы в правой части рабочей области отображается ее карточка с детальной информацией.

В верхнем правом углу карточки расположена кнопка действий (⋮). При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранной группы.

Карточка группы агентов содержит следующие вкладки:

- **Инфо** — основная информация о группе агентов.
- **Агенты** — список агентов в группе.

В нижней части карточки содержится кнопка **Конфигурация**, по которой открывается страница настройки конфигурации группы агентов.

Работа с группой агентов

Доступные операции над группами агентов:

- Создание группы агентов
- Просмотр группы агентов
- Дублирование группы агентов
- Удаление группы агентов

Создание группы агентов

Чтобы создать группу агентов:

1. Перейдите в раздел **Агенты** → **Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на кнопку **Создать** (+) на панели инструментов. Система отобразит окно создания группы.

Кнопка доступна, только если ни одна группа агентов из списка не выбрана.

3. Задайте название группы.
4. Если вы работаете в основном тенанте, выберите из выпадающего списка название тенанта, к которому относится группа.

Если у тенанта нет дочерних тенантов, он выбирается системой автоматически. Это поле отредактировать нельзя.

5. Если после создания группы вы хотите перейти к настройке ее конфигурации, позволяющей отслеживать различные действия на агентах, установите флажок **Перейти к конфигурации**.
6. Нажмите на кнопку **Создать**. Система создаст пустую группу и отобразит соответствующее уведомление. Созданная группа отобразится в списке раздела.

Наполнение пустой группы агентами выполняется в разделе **Агенты → Агенты**. Для этого доступны следующие операции:

- Добавление агента в группу
- Удаление агента из группы

Просмотр группы агентов

Чтобы просмотреть группу агентов:

1. Перейдите в раздел **Агенты → Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы агентов с подробной информацией о ней.

Название группы агентов отображается в верхней части карточки.

Карточка группы агентов содержит следующие вкладки:

- **Инфо** — основная информация о группе агентов.
- **Агенты** — список агентов в группе.

В нижней части карточки содержится кнопка **Конфигурация**, по которой открывается страница настройки конфигурации группы агентов.

Вкладка Инфо

На вкладке **Инфо** отображается основная информация о группе. Вкладка содержит следующие поля:

- **Название** — название группы агентов, используемое для ее идентификации

С помощью кнопки название можно изменить.

- **Тенант** — название тенанта, к которому относится группа.
- **Количество агентов** — количество агентов в группе.
- **Автоназначение** — индикатор включения автоматического назначения группы агентам (включено или выключено).
- **Условие** — VRL-выражение для автоматического назначения группы агентам.
- **Дата создания** — дата и время создания группы.
- **Дата изменения** — дата и время последнего изменения группы.

Вкладка Агенты

На вкладке **Агенты** отображается список агентов, входящих в группу. Список представляет собой таблицу со следующими столбцами:

- **Название** — название агента.
- **IP-адрес** — список IP-адресов узлов на агенте.

Для быстрого поиска агентов в списке введите запрос в строку поиска, расположенную над таблицей. В списке отобразятся агенты, в названии или IP-адресе которых встречается введенный запрос.

Чтобы получить подробные сведения об агенте, нажмите на его название в списке. Система откроет раздел **Агенты** → **Агенты** и отобразит на экране подробную карточку этого агента.

Дублирование группы агентов

Группы дублируются вместе с входящими в них агентами и настроенной конфигурацией.

Чтобы создать копию группы агентов:

1. Перейдите в раздел **Агенты** → **Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы агентов с подробной информацией о ней.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Дублировать**. Отобразится окно создания копии группы.
4. Задайте название копии группы.
5. Если после создания группы вы хотите перейти к настройке ее конфигурации, позволяющей отслеживать различные действия на агентах, установите флажок **Перейти к конфигурации**.
6. Нажмите на кнопку **Дублировать**. Система создаст копию выбранной группы и отобразит соответствующее уведомление. Созданная группа отобразится в списке раздела.

Создать копию группы можно также из экрана настройки ее конфигурации. Для этого:

1. Перейдите в раздел **Агенты** → **Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы агентов с подробной информацией о ней.
3. Нажмите на кнопку **Конфигурация** в нижней части карточки. Откроется страница настройки конфигурации выбранной группы.
4. Нажмите на кнопку на панели инструментов. Отобразится окно создания копии группы.
5. Задайте название копии группы.

6. Если после создания группы вы хотите перейти к настройке ее конфигурации, позволяющей отслеживать различные действия на агентах, установите флажок **Перейти к конфигурации**.
7. Нажмите на кнопку **Дублировать**. Система создаст копию выбранной группы и отобразит соответствующее уведомление. Созданная группа отобразится в списке раздела.

Удаление группы агентов

Чтобы удалить группу агентов:

1. Перейдите в раздел **Агенты** → **Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы агентов с подробной информацией о ней.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления группы.
4. Нажмите на кнопку **Удалить**. Система удалит группу и отобразит соответствующее уведомление. Удаленная группа будет исключен из списка раздела.

Удалить группу можно также из экрана настройки ее конфигурации. Для этого:

1. Перейдите в раздел **Агенты** → **Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы агентов с подробной информацией о ней.
3. Нажмите на кнопку **Конфигурация** в нижней части карточки. Откроется страница настройки конфигурации выбранной группы.
4. Нажмите на кнопку на панели инструментов. Отобразится окно подтверждения удаления группы.
5. Нажмите на кнопку **Удалить**. Система удалит группу и отобразит соответствующее уведомление. Удаленная группа будет исключен из списка раздела.

Вы также можете удалить сразу несколько групп агентов. Для этого:

1. Перейдите в раздел **Агенты** → **Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. В левом столбце таблицы установите флажки напротив тех групп, которые вы хотите удалить.
3. Нажмите на кнопку на панели инструментов. Отобразится окно подтверждения удаления групп.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные группы агентов и отобразит соответствующее уведомление. Удаленные группы будут исключены из списка раздела.

Настройка конфигурации группы агентов

Конфигурация группы агентов позволяет задать политики отслеживания действий на агентах в рамках группы.

Доступные операции над конфигурацией группы агентов:

- Просмотр конфигурации группы
- Настройка конфигурации группы

Просмотр конфигурации группы

Чтобы просмотреть текущую конфигурацию группы агентов:

1. Перейдите в раздел **Агенты** → **Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы агентов с подробной информацией о ней.
3. Нажмите на кнопку **Конфигурация** в нижней части карточки. Откроется страница настройки конфигурации выбранной группы.

Вы также можете открыть страницу конфигурации при создании группы на базе агента, создании пустой группы, а также дублировании группы путем установки флажка **Перейти к конфигурации**.

Панель инструментов страницы включает в себя следующие компоненты:

- Кнопка **Удалить** ( ) позволяет удалить выбранную группу агентов.
- Кнопка **Дублировать** ( ) позволяет создать копию выбранной группы агентов.

На рабочей области страницы отображаются параметры настройки конфигурации, которые сгруппированы по разделам, расположенным слева. Рядом с параметрами конфигурации присутствуют переключатели, позволяющие их активировать.

При активации параметров отображаются их настройки, а также кнопки для сохранения заданных настроек или отмены несохраненных изменений.

Настройка конфигурации группы

Чтобы настроить конфигурацию группы агентов:

1. Перейдите в раздел **Агенты** → **Группы агентов**. Система отобразит сведения об имеющихся группах агентов.
2. Нажмите на строку группы в списке. Система отобразит в правой части экрана карточку этой группы агентов с подробной информацией о ней.
3. Нажмите на кнопку **Конфигурация** в нижней части карточки. Откроется страница настройки конфигурации выбранной группы.

Вы также можете открыть страницу конфигурации при создании группы на базе

агента, создании пустой группы, а также дублировании группы путем установки флажка **Перейти к конфигурации**.

4. Задайте настройки конфигурации группы агентов.

Конфигурацию группы агентов следует настраивать аккуратно. Неправильная настройка конфигурации может привести к избыточной нагрузке на агентах и существенно увеличить количество передаваемых событий.

В системе доступна настройка следующих параметров:

- **Автоназначение** — автоматическое назначение группы агенту.
- **Пользователи и процессы**:
  - **Вход/выход пользователей** — отслеживание начала и окончания сеансов пользователей.
  - **Создание процессов** — отслеживание процессов, запущенных пользователями и системой.
  - **Добавление или изменение сервисов** — отслеживание изменений в файлах сервисов.
  - **Управление пользователями** — отслеживание действий по управлению пользователями.
  - **Не использовать auditd** — использование только eBPF для сбора событий.
- **Чтение файлов** — сбор данных журнала из файлов и событий Windows.
- **Выполнение команд** — сбор данных журнала из вывода команд.
- **WatchDog** — отслеживание процессов на агенте.
- **Доступ к объектам** — отслеживание действий с объектами в указанных путях.

После настройки любого параметра необходимо нажать кнопку **Сохранить**.

- Если в описании параметра указаны названия ОС, включенный параметр будет применен именно к ним.
- Если в описании параметра не указаны названия ОС, включенный параметр будет применен ко всем ОС.

Автоназначение

Позволяет указать выражение на языке преобразования данных VRL, определяющее условия автоматического назначения группы агенту.

Чтобы использовать автоназначение:

1. Установите переключатель **Автоназначение** в активное положение. Отобразится поле для ввода VRL-выражения.
2. Укажите выражение на языке VRL, по которому группа будет автоматически назначаться агентам.
3. Нажмите на кнопку **Сохранить**.

Система будет сопоставлять заданное выражение с данными сканирования агента. При совпадении данных агент будет включен в текущую группу.

Автоматическое назначение группы происходит только при регистрации новых агентов. Для уже существующих в системе агентов автоназначение не выполняется.

Если данные сканирования агента будут удовлетворять выражению сразу в нескольких группах агентов, система включит агент во все эти группы.

▼ *Пример автоназначения группы*

Пример выражения для автоматического назначения группы агентам с адресом, входящим в подсеть 10.99.66.0/24:

```
# Проверяем каждый адрес каждого интерфейса на вхождение в подсеть.

subnetwork = "10.99.66.0/24"

networks = values(object!(.networkInterfaces.values))
ipv4s = flatten(map_values(networks) -> |value| { value.ipv4 })
addresses = compact(map_values(ipv4s) -> |value| { value.address })

is_in_subnetwork = length(filter(addresses) -> |_index, value| {
  ip_cidr_contains!(subnetwork, value)
}) > 0

is_in_subnetwork
```

Пользователи и процессы

Параметры раздела **Пользователи и процессы**:

- Вход/выход пользователей (Linux/macOS)
- Создание процессов (Linux/macOS)
- Добавление или изменение сервисов (Linux)
- Управление пользователями (Linux/macOS)
- Не использовать auditd (Linux)

Вход/выход пользователей (Linux/macOS)

Этот параметр отслеживает начало и окончание сеансов пользователей в системе.

Параметр работает в ОС Linux и macOS.

Чтобы включить параметр, установите переключатель **Вход/выход пользователей (Linux/macOS)** в активное положение и нажмите на кнопку **Сохранить**.

Создание процессов (Linux/macOS)

Этот параметр отслеживает процессы, запущенные пользователями или системой.

Параметр работает в ОС Linux и macOS.

Чтобы включить параметр, установите переключатель **Создание процессов (Linux/macOS)** в активное положение и нажмите на кнопку **Сохранить**.

Добавление или изменение сервисов (Linux)

Этот параметр отслеживает добавление и изменение сервисных файлов .systemd, .timer и rc.

Параметр работает в ОС Linux.

Чтобы включить параметр, установите переключатель **Добавление или изменение сервисов (Linux)** в активное положение и нажмите на кнопку **Сохранить**.

Управление пользователями (Linux/macOS)

Этот параметр отслеживает изменения пользователей, групп пользователей, прав sudo и настроек окружения (bashrc).

Параметр работает в ОС Linux и macOS.

Чтобы включить параметр, установите переключатель **Управление пользователями (Linux)** в активное положение и нажмите на кнопку **Сохранить**.

Не использовать auditd (Linux)

Этот параметр позволяет отказаться от использования AuditD и собирать ИБ-события только через eBPF. Это может быть полезно при работе в Linux-системах с актуальными версиями ядра, а также в ситуациях, когда использование AuditD нежелательно.

Параметр работает в ОС Linux.

Чтобы включить параметр, установите флажок **Не использовать auditd (Linux)** и нажмите на кнопку **Сохранить**.

Чтение файлов

Этот параметр используется для настройки сбора данных журнала из файлов и событий ОС.

Значимые события (например, ошибки при работе приложений) записываются системой, под которой работает агент, в специальные журналы событий. Также данные можно собирать из файлов в системе агента.

При настройке параметра требуется указать, каким способом и по какому пути агенты текущей группы должны собирать данные.

Чтобы настроить параметр:

1. Установите переключатель **Чтение файлов** в активное положение. Система отобразит настройки сбора данных журнала.
2. Задайте настройки сбора данных.

3. Нажмите на кнопку **Сохранить**. Система сохранит заданные настройки сбора данных с агентов.

Добавление настройки сбора данных

Чтобы добавить настройку сбора данных в параметре **Чтение файлов**:

1. Нажмите на кнопку **Добавить настройку** (+). Отобразится строка с набором полей для настройки нового типа журнала.
2. Выберите тип журнала из выпадающего списка и задайте его настройки.

   Доступные типы журналов:

   - **audit** — используется для событий из инструмента Linux Audit Daemon (auditd). Объединяет последовательные журналы с одинаковым идентификатором в одно событие.
   - **djb-multilog** — используется для чтения файлов в формате, созданном сервисным регистратором с несколькими журналами в daemontools.
   - **eventchannel** — используется для журналов событий Windows, получает события в формате JSON, контролирует все каналы, указанные в файле конфигурации, и показывает все включенные в них поля. Может использоваться для мониторинга стандартных журналов событий Windows, а также журналов приложений и служб.

     При выборе этого типа становятся доступны следующие настройки:

     - **Имя журнала** — имя журнала событий Windows.
     - **Фильтр** — запрос на языке XPath, позволяющий указать критерии, по которым будут отфильтрованы собираемые данные.

       Например, запрос вида `Event/System[EventID != 5145 and EventID != 5156]` позволит собрать из папки `event/system` в системе, где установлен агент, все события, за исключением событий с идентификаторами 5145 и 5156.
   - **eventfile** — используется для сбора событий Windows из файлов формата EVTX.

     При выборе этого типа становятся доступна настройка **Фильтр** — запрос на языке XPath, позволяющий указать критерии, по которым будут отфильтрованы собираемые данные.

     Например, запрос вида `Event/System[EventID != 5145 and EventID != 5156]` позволит собрать из папки `event/system` в системе, где установлен агент, все события, за исключением событий с идентификаторами 5145 и 5156.
   - **eventlog** — используется для классического формата журнала событий Windows.
   - **iis** — используется для IIS (веб-сервера Windows).
   - **json** — используется для однострочных JSON-файлов и позволяет добавлять настраиваемые метки к JSON-событиям.
   - **macos** — используется для журналов системы логирования macOS ULS и получает журналы в формате системного журнала.
   - **mysql_log** — используется для журналов MySQL. Не поддерживает многострочные журналы.
   - **nmapg** — используется для мониторинга файлов, соответствующих выводу данных в grepable

формате при использовании утилиты nmap.

- **postgresql_log** — используется для журналов PostgreSQL. Не поддерживает многострочные журналы.
- **snort-full** — используется для формата полного вывода системы обнаружения вторжений Snort.
- **squid** — используется для журналов программного пакета squid.
- **syslog** — используется для простых текстовых файлов в формате, подобном системному журналу.

Для каждого типа журнала, кроме **eventchannel**, нужно указать путь, используемый системой при сборе данных на агенте, в поле **Путь**.

Удаление настройки сбора данных

Чтобы удалить настройки журнала, нажмите на кнопку справа от его строки настроек.

Выполнение команд

Этот параметр используется для вывода команд, позволяющих записывать значимые события (например, ошибки при работе приложений) системой, под которой работает агент, в специальные журналы событий.

При настройке параметра требуется указать частоту использования команды и ее текст.

Чтобы настроить параметр:

1. Установите переключатель **Выполнение команд** в активное положение. Система отобразит настройки выполнения команд.
2. Задайте настройки выполнения команд. Для этого:
3. Нажмите на кнопку **Сохранить**. Система сохранит заданные настройки выполнения команд.

Добавление настройки выполнения команд

Чтобы добавить настройку выполнения команд:

1. Нажмите на кнопку **Добавить настройку** (+). Отобразится строка с набором полей для настройки нового типа журнала.
2. Выберите тип журнала из выпадающего списка и задайте его настройки.

   Доступные типы журналов:

   - **command** — используется для чтения вывода команды, запущенной пользователем root и указанной в теге команды. Каждая строка вывода обрабатывается как отдельный журнал.

     При выборе этого типа становятся доступны настройки:

     - **Частота** — частота использования команды (сек.).
     - **Команда** — текст команды. Например, команда `last -n 5` отобразит список последних 5 терминальных сессий.

- **full_command** — используется для чтения вывода команды, запущенной пользователем root и указанной в теге команды. Весь вывод будет рассматриваться как один элемент журнала.

  При выборе этого типа становятся доступны настройки:

  - **Частота** — частота использования команды (сек.).
  - **Команда** — текст команды.

Удаление настройки выполнения команд

Чтобы удалить настройки журнала, нажмите на кнопку справа от его строки настроек.

WatchDog

Этот параметр запускает сервис WatchDog, предназначенный для отслеживания процессов на агенте. При работающем сервисе остановка контролируемых процессов недоступна.

Чтобы включить и настроить параметр:

1. Установите переключатель **WatchDog** в активное положение.
2. По умолчанию отключение сервиса WatchDog выполняется без пароля. Чтобы при попытке отключить сервис запрашивался пароль, установите флажок **Пароль** и задайте нужный пароль в появившемся поле.
3. Нажмите на кнопку **Сохранить**. Система сохранит заданные настройки сервиса WatchDog.

Доступ к объектам (Linux)

Этот параметр отслеживает действия с объектами в указанных путях. Отслеживаются чтение, запись и запуск объектов, а также изменение прав доступа к ним.

Политика работает в ОС Linux.

Чтобы включить и настроить параметр:

1. Установите переключатель **Доступ к объектам (Linux)** в активное положение.
2. Задайте перечень путей к объектам. Для этого:
   a. Укажите путь и задайте права доступа к объектам, хранящимся по этому пути, посредством установки соответствующих флажков:
      - **w (Writing)** — запись;
      - **r (Reading)** — чтение;
      - **x (Execution)** — выполнение;
      - **a (Attributes)** — атрибуты.
   b. Чтобы добавить следующий путь, нажмите на кнопку **Добавить путь** (+). Появится строка настройки нового пути.
   c. Чтобы удалить ненужные пути, нажмите на кнопку справа от их строк.
3. Задайте при необходимости шаблоны исключений в секции **Исключения**:

- **По пути** — исключают из рассмотрения указанные пути.
- **По имени файлов** — исключают из рассмотрения файлы с указанными именами.
- **По пути родительского процесса** — исключают из рассмотрения указанные пути родительского процесса.

При задании шаблонов исключений поддерживается использование символа-подстановки (*). Например, исключение по имени файлов вида `*.log` позволит исключить из рассмотрения все файлы с расширением `.log`.

4. Нажмите на кнопку **Сохранить**. Система сохранит заданные настройки доступа к объектам.

Синтаксис языка XPath

При настройке конфигурации групп агентов в параметре **Чтение файлов** доступно задание фильтров на языке XPath.

- Операторы и специальные символы
- Коллекции
- Операторы сравнения
- Логические выражения
- Выражение набора
- Функции
- Оси

Операторы и специальные символы

Обозначение	Описание
`/`	Дочерний элемент.
`/descendant-or-self::node()` `//`	Рекурсивный спуск.
`self::node()` `.`	Текущий контекст.
`parent::node()` `..`	Родитель узла текущего контекста.
`child::*` `*`	Все элементы.
`@`	Префикс имени атрибута.
`attribute::*` `@*`	Все атрибуты.
`:`	Разделитель пространства имен.

Обозначение	Описание
`( )`	Группировка операций.
`[ ]`	Шаблон фильтра.
`+`	Сложение.
`-`	Вычитание.
`div`	Деление с плавающей запятой.
`*`	Умножение.
`mod`	Остаток от деления.

Коллекции

Обозначение	Описание
`author[1]`	Первый элемент `<author>`.

Операторы сравнения

Обозначение	Описание
`<`	Меньше.
`<=`	Меньше или равно.
`=`	Равно.
`>`	Больше.
`>=`	Больше или равно.

Логические выражения

Обозначение	Описание
`and`	Логическое "И".
`or`	Логическое "ИЛИ".
`not()`	Отрицание.

Выражение набора

Обозначение	Описание
`|`	Операция множества для объединения узлов.

Функции

- Функции для наборов узлов
- Строковые функции
- Логические функции
- Численные функции

Функции для наборов узлов

Обозначение	Описание
`count()`	Получение количества узлов в множестве.
`id()`	Выбор элемента с уникальным идентификатором.
`last()`	Получение номера последнего элемента в множестве.
`local-name()`	Получение имени первого узла в множестве.
`name()`	Получение полного имени узла в множестве.
`namespace-uri()`	Получение URI-кода пространства имен.
`position()`	Получение позиции элемента в множестве.

Строковые функции

Обозначение	Описание
`concat(string, string, string*)`	Сцепление строк
`contains(string, string)`	Определение, есть ли в заданной строке искомая строка.
`normalize-space(string)`	Удаление начальных, конечных и повторяющихся пробелов в строке.
`starts-with(string, string)`	Определение, есть ли в начале строки искомая строка.
`string(object)`	Преобразование объекта в строку.
`string-length(object)`	Получение числа символов в строке.
`substring(string, number, number?)`	Получение подстроки.
`substring-after(string, string)`	Получение подстроки, которая следует за вторым аргументом строки.
`substring-before(string, string)`	Получение подстроки, которая следует до второго аргумента строки.
`translate(string, string, string)`	Получение строки, в которой символы из второго аргумента заменены на символы третьего аргумента с учетом позиции.

Логические функции

Обозначение	Описание
`boolean()`	Преобразование аргумента в логическое значение.
`false()`	Получение значения `false`.
`lang(string)`	Определение, совпадает ли атрибут `xml:lang` со строкой аргумента.
`not()`	Отрицание логического значения.
`true()`	Получение значения `true`.

Численные функции

Обозначение	Описание
`ceiling()`	Округление в большую сторону.
`floor()`	Округление в меньшую сторону.
`number()`	Преобразование аргумента в число.
`round()`	Округление до ближайшего целого числа.
`sum()`	Получение суммы всех узлов в множестве.

Оси

Обозначение	Описание
`ancestor::`	Предки узла контекста.
`ancestor-or-self::`	Узел контекста и его предки.
`@ attribute`	Атрибуты контекстного узла.
`child::`	Дочерние элементы узла контекста.

`.//`
 	
`descendant`
	Потомки контекстного узла.

`descendant-or-self::`	Узел контекста и его потомки.
`following::`	Все узлы, расположенные на дереве после узла контекста.
`following-sibling::`	Все следующие элементы узла контекста с общим родителем.
`namespace::`	Узлы пространства имен узла контекста.

`..`
 	
`parent`
	Родитель узла контекста.

`preceding::`	Все узлы, расположенные на дереве перед узлом контекста.
`preceding-sibling::`	Все предшествующие элементы узла контекста с общим родителем.

`.`
 	
`self`
	Только сам контекстный узел.

Работа со шлюзами

Данный раздел описывает процесс работы со шлюзами в системе. Работа осуществляется в разделе **Агенты** → **Шлюзы** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О шлюзах

Шлюз представляет собой промежуточный уровень в сети между агентами, установленными на конечных устройствах, и центральным кластером.

Шлюзы предназначены для управления потоками данных между агентами и центральным кластером. Они позволяют настраивать маршрутизацию, агрегацию и фильтрацию данных.

Подробности подключения агента к кластеру с использованием шлюзов описаны в разделе Подключение агента к кластеру.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новый шлюз.
- Кнопка позволяет обновить список шлюзов.
- Кнопка позволяет удалить выбранные шлюзы.
- Поле **Поиск** предназначено для быстрого поиска шлюзов в отображаемом списке по названию и пространству.
- Кнопка **Фильтр** ( ) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка шлюзов.

В рабочей области раздела отображается таблица имеющихся шлюзов. Таблица содержит следующие столбцы:

- **Название** — название шлюза, используемое для его идентификации.
- **Статус** — текущий статус шлюза:
  - **В очереди** — шлюз находится в очереди на запуск.
  - **Запускается** — шлюз находится в процессе запуска.
  - **Запущен** — шлюз запущен и готов к подключениям.
  - **Останавливается** — шлюз находится в процессе остановки.
  - **Остановлен** — шлюз остановлен и недоступен для взаимодействия.
  - **Ошибка** — ошибка размещения шлюза в пространстве.
  - **Неизвестно** — неизвестное состояние шлюза.
- **Пространство** — название подключения к пространству, в котором создан шлюз.

При работе с таблицей шлюзов доступны следующие операции:

- Поиск шлюзов по полям **Название** и **Пространство**.
- Фильтрация шлюзов по полю **Пространство**. Пространство выбирается из выпадающего списка.
- Сортировка шлюзов по полям **Название** и **Пространство**.
- Настройка отображения таблицы шлюзов.

При выборе конкретного шлюза в правой части рабочей области отображается его карточка с детальной информацией.

В верхнем правом углу карточки расположена кнопка для удаления шлюза.

Карточка содержит основную информацию о шлюзе.

Работа со шлюзом

Доступные операции над шлюзом:

- Создание шлюза
- Просмотр шлюза
- Изменение шлюза
- Удаление шлюза
- Запуск шлюза
- Остановка шлюза

Создание шлюза

Чтобы создать шлюз:

1. Перейдите в раздел **Агенты** → **Шлюзы**. Система отобразит сведения об имеющихся шлюзах.
2. Нажмите на кнопку **Создать** (+) на панели инструментов. Система отобразит окно создания шлюза.
3. Задайте название шлюза.
4. Выберите подключение к пространству, в котором необходимо создать шлюз.
5. Укажите IP- или FQDN-адрес для подключения к шлюзу.

Адреса, используемые для подключения к шлюзу, зависят от типа кластера, в котором создается шлюз.

- Если шлюз создается в пространстве центрального управляющего кластера, используйте IP-адреса любого узла центрального кластера.
- Если шлюз создается в пространстве кластера, являющегося сателлитом, используйте IP-адреса любого узла сателлита.

FQDN-адрес можно использовать, только если в инфраструктуре существует DNS-сервер и администратор настроил прямую зону (forward zone). Проверить, настроена ли такая зона, можно с помощью утилиты nslookup:

```
nslookup <domain_name>
```

Здесь:

- `<domain_name>` — доменное имя.

Обратные DNS-запросы для определения FQDN-адресов по IP-адресам возможны, если на DNS-сервере настроена обратная зона (reverse zone). В ней хранятся записи, сопоставляющие IP-адреса с доменными именами. Проверить, настроена ли такая зона, можно также с помощью утилиты nslookup:

```
nslookup <IP_address>
```

Здесь:

- `<IP_address>` — IP-адрес узла.

Если обратная зона не настроена, свяжитесь с сетевым администратором.

Если ваш компьютер имеет доступ к кластеру с установленным файлом конфигурации kubeconfig, вы можете получить сведения обо всех IP-адресах узлов кластера. Для сбора IP-адресов всех узлов в кластере и их вывода в виде строки используйте команду:

```
kubectl get node -o json
--output='jsonpath={.items[*].status.addresses[0].address}'
```

6. Укажите порт для подключения к шлюзу. Допустимые значения: 30 000—32 767.

Порт должен быть уникальным в рамках инсталляции системы.

7. Задайте максимальный размер буфера в МБ для хранения событий ИБ.

Эффективный размер буфера будет равен 80% от указанного значения.

8. При необходимости введите описание шлюза.
9. При необходимости шифрования потока данных между агентом и шлюзом через TLS переведите переключатель **Включить TLS** в активное положение.

   При включении TLS необходимо выбрать секрет типа **Контейнер PKCS**, хранящий данные сертификата TLS.

Выбранный секрет запоминается системой и сохраняется в шлюзе. Таким образом, в случае удаления секрета шлюз продолжит работать без ошибок, даже если будет выключен и включен заново.

10. Нажмите на кнопку **Создать**. Система создаст шлюз и отобразит соответствующее уведомление. Созданный шлюз отобразится в списке раздела и будет запущен.

Просмотр шлюза

Чтобы просмотреть шлюз:

1. Перейдите в раздел **Агенты** → **Шлюзы**. Система отобразит сведения об имеющихся шлюзах.

2. Нажмите на строку шлюза в списке. Система отобразит в правой части экрана карточку этого шлюза с подробной информацией о нем.

Название шлюза отображается в верхней части карточки.

Карточка содержит следующие сведения о шлюзе:

- **ID шлюза** — уникальный идентификационный код шлюза, генерируемый автоматически при его создании.
- **Название** — имя шлюза, используемое для его идентификации в системе.
- **Описание** — краткое описание функциональности шлюза.
- **Статус** — текущий статус шлюза:
  - **В очереди**;
  - **Запускается**;
  - **Запущен**;
  - **Останавливается**;
  - **Остановлен**;
  - **Ошибка**;
  - **Неизвестно**.
- **IP-адрес или FQDN** — IP- или FQDN-адрес для подключения к шлюзу.
- **Порт** — порт для подключения шлюза к другим системам.
- **Пространство** — название подключения к пространству, в котором создан шлюз.
- **Размер стрима для событий ИБ** — максимальный размер буфера для хранения событий ИБ. Если размер буфера меньше 1024 МБ, то значение отображается в МБ, иначе — в ГБ.
- **Шифрование** — текущий статус шифрования через TLS. Доступные значения:
  - **Включено** — шифрование соединения к шлюзу через TLS включено.
  - **Выключено** — шифрование соединения к шлюзу через TLS выключено.
- **Секрет** — секрет, использующийся для TLS-шифрования.

  Чтобы просмотреть секрет на странице раздела **Секреты**, нажмите на кнопку **Открыть**.

Изменение шлюза

Чтобы изменить название или описание шлюза:

1. Перейдите в раздел **Агенты** → **Шлюзы**. Система отобразит сведения об имеющихся шлюзах.
2. Нажмите на строку шлюза в списке. Система отобразит в правой части экрана карточку этого шлюза с подробной информацией о нем.
3. Нажмите на кнопку рядом с текущим названием или описанием шлюза в его карточке. Система отобразит поле ввода значения.
4. Введите новое значение поля.
5. Нажмите **Сохранить**. Система сохранит изменения и отобразит соответствующее уведомление.

Обновленная информация отобразится в карточке шлюза и в списке раздела.

Удаление шлюза

Чтобы удалить шлюз:

1. Перейдите в раздел **Агенты** → **Шлюзы**. Система отобразит сведения об имеющихся шлюзах.
2. Нажмите на строку шлюза в списке. Система отобразит в правой части экрана карточку этого шлюза с подробной информацией о нем.
3. Нажмите на кнопку в верхнем правом углу карточки. Отобразится окно подтверждения удаления шлюза.
4. Нажмите на кнопку **Удалить**. Система удалит шлюз и отобразит соответствующее уведомление. Удаленный шлюз будет исключен из списка раздела.

Вы также можете удалить сразу несколько шлюзов. Для этого:

1. Перейдите в раздел **Агенты** → **Шлюзы**. Система отобразит сведения об имеющихся шлюзах.
2. В левом столбце таблицы установите флажки напротив тех шлюзов, которые вы хотите удалить.
3. Нажмите на кнопку на панели инструментов. Отобразится окно подтверждения удаления шлюзов.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные шлюзы и отобразит соответствующее уведомление. Удаленные шлюзы будут исключены из списка раздела.

Запуск шлюза

Чтобы запустить остановленный шлюз:

1. Перейдите в раздел **Агенты** → **Шлюзы**. Система отобразит сведения об имеющихся шлюзах.
2. Нажмите на строку шлюза в списке. Система отобразит в правой части экрана карточку этого шлюза с подробной информацией о нем.
3. Переведите переключатель в верхней части вкладки в активное положение. Система запустит шлюз и отобразит соответствующее уведомление. Шлюз перейдет в статус **Запущен**.

Остановка шлюза

Чтобы остановить запущенный шлюз:

1. Перейдите в раздел **Агенты** → **Шлюзы**. Система отобразит сведения об имеющихся шлюзах.
2. Нажмите на строку шлюза в списке. Система отобразит в правой части экрана карточку этого шлюза с подробной информацией о нем.
3. Переведите переключатель в верхней части вкладки в неактивное положение. Система остановит шлюз и отобразит соответствующее уведомление. Шлюз перейдет в статус **Остановлен**.

Сканирование на уязвимости

В этом разделе приведены инструкции по настройке и выполнению сканирования оборудования на наличие уязвимостей.

Перед проведением сканирования выполните подготовительные действия:

1. Загрузите в систему базу уязвимостей. Это можно сделать вручную, загрузив файл, или настроить автоматическое обновление базы уязвимостей.
2. Создайте сканеры в пространствах.
3. Настройте учетные записи, которые будут использоваться для аутентификации на оборудовании в процессе его сканирования и поиска уязвимостей.
4. Настройте профили сканирования, содержащие параметры сканирования оборудования и сетей.
5. Создайте задачу сканирования оборудования и сетей и определите периодичность ее выполнения. Задача использует данные профиля и учетной записи, которые указываются при ее создании.

База уязвимостей R-Vision

С информацией о системах, которые покрывает база уязвимостей R-Vision в профилях сканирования, вы можете ознакомиться на справочном портале R-Vision.

Интеграции с внешними системами

Данный раздел описывает процесс настройки интеграций с внешними системами при сканировании. Настройка осуществляется в разделе **Автоматизация** → **Запуски** веб-интерфейса системы.

Чтобы настроить интеграцию с внешней системой:

1. Создайте логин и пароль учетной записи интеграции.
2. Выполните общую настройку интеграции и укажите данные учетной записи.
3. Выполните настройку конкретной внешней системы:
   - Kaspersky Security Center
   - Zabbix
   - VMWare vCenter
   - Active Directory
4. При необходимости выполните настройку расписания запуска интеграции.

Создание учетной записи интеграции

Для создания учетной записи используется раздел **Переменные**. Для каждой интеграции нужно создать две переменные. Одна из них будет использоваться как логин, а другая — как пароль.

Чтобы создать логин учетной записи интеграции:

1. Перейдите в раздел **Ресурсы** → **Переменные**. Система отобразит сведения об имеющихся переменных.
2. Нажмите на кнопку **Добавить** (＋). Отобразится окно создания переменной.
3. Введите человекочитаемый идентификатор записи логина.
4. Введите название записи логина.
5. При необходимости введите описание логина.
6. Выберите из выпадающего списка тип переменной **String**. Отобразится поле для ввода значения переменной.
7. Переведите переключатель **Секрет** в активное положение.
8. В поле **Значение переменной** введите логин.
9. Нажмите **Добавить**.

Теперь выполните шаги 2—9 для создания пароля учетной записи интеграции.

Общая настройка

Чтобы выполнить общую настройку интеграции:

1. Перейдите в раздел **Автоматизация** → **Запуски**. Система отобразит сведения об имеющихся запусках.
2. Нажмите на кнопку **Запустить** (▷). Отобразится окно выбора элемента экспертизы для запуска.
3. Из раскрывающегося списка **Тип** выберите тип элементов экспертизы **Плейбук**.
4. Раскройте древо каталогов в левой части окна и выберите каталог **Интеграции**, содержащий плейбуки с интеграциями.
5. Выберите в каталоге **Интеграции** элемент с названием внешней системы для интеграции. В правой части окна отобразится плейбук интеграции с этой внешней системой.
6. Выберите плейбук запуска интеграции.
7. Нажмите на кнопку **Добавить**. Выбранный плейбук будет добавлен в запуск. После добавления выбранного плейбука в запуск открывается окно параметров запуска. Перейдите к заполнению данных в блоке **Аргументы**.

Чтобы указать логин и пароль в параметрах интеграции, нажмите на кнопку {;} и выберите переменную, которая содержит эти данные.

Интеграция с Kaspersky Security Center

При настройке интеграции с Kaspersky Security Center:

1. Настройте учетную запись.
2. Заполните необходимые поля и активируйте необходимые переключатели в блоке **Аргументы**.

Настройка учетной записи для интеграции с Kaspersky Security Center

Для импорта информации из Kaspersky Security Center системе потребуется учетная запись с

правами на чтение базы данных Kaspersky Security Center.

Для настройки учетной записи с ограниченными правами выполните следующие действия:

1. В дереве консоли выберите узел с именем нужного вам Сервера администрирования.
2. В контекстном меню Сервера администрирования выберите пункт **Свойства**.
3. В окне свойств Сервера администрирования выберите раздел **Безопасность**.
4. В поле **Имена групп или пользователей** выберите пользователя, которому нужно присвоить роль.
5. Если пользователь отсутствует в поле, добавьте его с помощью кнопки **Добавить**.
6. В списке прав в нижней части окна, на закладке **Права** настройте права для пользователя:
   a. Нажмите на кнопку со значком "плюс" (+), чтобы раскрыть все доступные узлы в списке, и назначьте права **Чтение**.
   b. Установите флажки **Разрешить** рядом с правами **Чтение**.

Учетная запись настроена.

Для настройки учетной записи вы можете не настраивать права, а присвоить пользователю одну из системных ролей оператора: например, **Главный оператор**, **Оператор Kaspersky Security Endpoint** или **Оператор Сервера администрирования**.

Чтобы присвоить роль пользователю:

1. Перейдите на закладку **Роли** и нажмите на кнопку **Добавить**. Откроется окно **Роли пользователей**. В окне отображаются созданные роли пользователей.
2. В окне **Роли пользователей** выберите роль для пользователя.
3. Нажмите на кнопку **ОК**.

Поля в блоке Аргументы

Аргумент	Значение	Редактируемое поле
`DeviceFields`	Дополнительные атрибуты технических характеристик устройств в KSC, которые необходимо импортировать. Список доступных атрибутов можно посмотреть в официальной документации KSC.	Да
`HostFields`	Дополнительные атрибуты устройств в KSC, которые необходимо импортировать. Список доступных атрибутов можно посмотреть в официальной документации KSC.	Да
`HostFilters`	Фильтр оборудования. Формат фильтра описан в подразделе Формат фильтра оборудования.	Да

Аргумент	Значение	Редактируемое поле
`HostGroupNames`	Фильтр по имени группы. Пример: Имя главной группы/Имя подгруппы	Да
`KSCHost`	Адрес сервера KSC.	Да
`KSCPassword`	Пароль учетной записи KSC.	Да
`KSCPort`	Порт подключения.	Да
`KSCProtocol`	Протокол подключения.	Да
`KSCUsername`	Логин учетной записи KSC.	Да
`LinksNormalizations`	Список ID правил нормализации для связей, которые необходимо импортировать: • Связь **Оборудование — УЗ**: ksc.isassets.rvision.ru.device_to_accounts • Связь **Оборудование — ПО**: ksc.isassets.rvision.ru.device_to_software • Связь **Оборудование — Обновления**: ksc.isassets.rvision.ru.device_to_update • Связь **Оборудование — Уязвимости**: ksc.vulnerabilities.rvision.ru.vulnToDevice • Связь **Уязвимости — ПО**: ksc.vulnerabilities.rvision.ru.vulnToSoftware	Да
`Source`	Тип источника данных.	Нет

Аргумент	Значение	Редактируемое поле
`EntitiesNormalizations`	Список ID правил нормализации для сущностей, которые необходимо импортировать: • **Оборудование**: ksc.host.isassets.rvision.ru.Device • **Обновления**: ksc.update.isassets.rvision.ru.Updates • **ПО**: ksc.software.isassets.rvision.ru.Software • **Учетные записи**: ksc.user.isassets.rvision.ru.Accounts • **Уязвимости**: ksc.vulnerability.vulnerabilities.rvision.ru.Vulnerabilities • **Уязвимое ПО**: ksc.affected_software.isassets.rvision.ru.Software При необходимости набор импортируемых полей и маппинг можно отредактировать в соответствующем правиле нормализации. Правила нормализации для интеграций располагаются в разделе **Экспертиза** в каталоге **Интеграции**.	Да
`UpdateFields`	Дополнительные атрибуты обновлений устройств в KSC, которые необходимо импортировать. Список доступных атрибутов можно посмотреть в официальной документации KSC.	Да
`UserFields`	Дополнительные атрибуты пользователей устройств в KSC, которые необходимо импортировать. Список доступных атрибутов можно посмотреть в официальной документации KSC.	Да

Аргумент	Значение	Редактируемое поле
`VulnerabilityFields`	Дополнительные атрибуты уязвимостей устройств в KSC, которые необходимо импортировать. Список доступных атрибутов можно посмотреть в официальной документации KSC.	Да

Формат фильтра оборудования

Подробная информация о синтаксисе фильтра представлена в официальной документации KSC.

Атрибуты фильтра:

- `KLHST_WKS_IP_LONG` — IP-адрес в формате десятичного числа. Например: 10.99.101.200 → 174286280.
- `KLHST_WKS_DN` — имя устройства

Операторы:

- Равно: `=`
- Больше: `>`
- Меньше: `<`
- Не равно: `<>`
- Логическое "И": `&`
- Логическое "ИЛИ": `|`
- Логическое "НЕ": `!`

Особенности составления условий фильтрации:

- Значения атрибутов указываются в кавычках.

  Пример 42. Пример фильтра по значению IP-адреса

  ```
  (KLHST_WKS_IP_LONG = "174286280")
  ```

- Оператор отрицания (логического "НЕ") указывается перед названием атрибута.

  Пример 43. Пример условия с отрицанием

  ```
  (!KLHST_WKS_DN = "DRWEBESS01")
  ```

- Операторы логических "И" и "ИЛИ" указываются в начале, перед операндами.

Пример 44. Пример фильтра по диапазону IP-адресов

```
(&(KLHST_WKS_IP_LONG >= "174286280")(KLHST_WKS_IP_LONG <= "174286328"))
```

Пример 45. Пример фильтра по IP-адресу и имени устройства

```
(|(KLHST_WKS_IP_LONG = "174286297")(KLHST_WKS_DN = "DRWEBESS01"))
```

Переключатели в блоке Аргументы

Аргумент	Значение	Редактируемое поле
`HostIncludeVisibleOnly`	Импортировать недоступные устройства.	Да
`IgnoreLastRun`	Игнорировать дату последнего запуска.	Да
`IncludeHostDevices`	Импортировать технические характеристики оборудования.	Да
`IncludeHostProducts`	Импортировать данные об установленных продуктах Kaspersky.	Да
`IncludeHostSoftware`	Импортировать ПО.	Да
`IncludeHostUpdates`	Импортировать обновления.	Да
`IncludeHostUsers`	Импортировать локальных пользователей.	Да
`IncludeHostVulnerabilities`	Импортировать уязвимости.	Да

Интеграция с Zabbix

При настройке интеграции с Zabbix:

1. Настройте учетную запись.
2. Заполните необходимые поля в блоке **Аргументы**.

Настройка учетной записи для интеграции с Zabbix

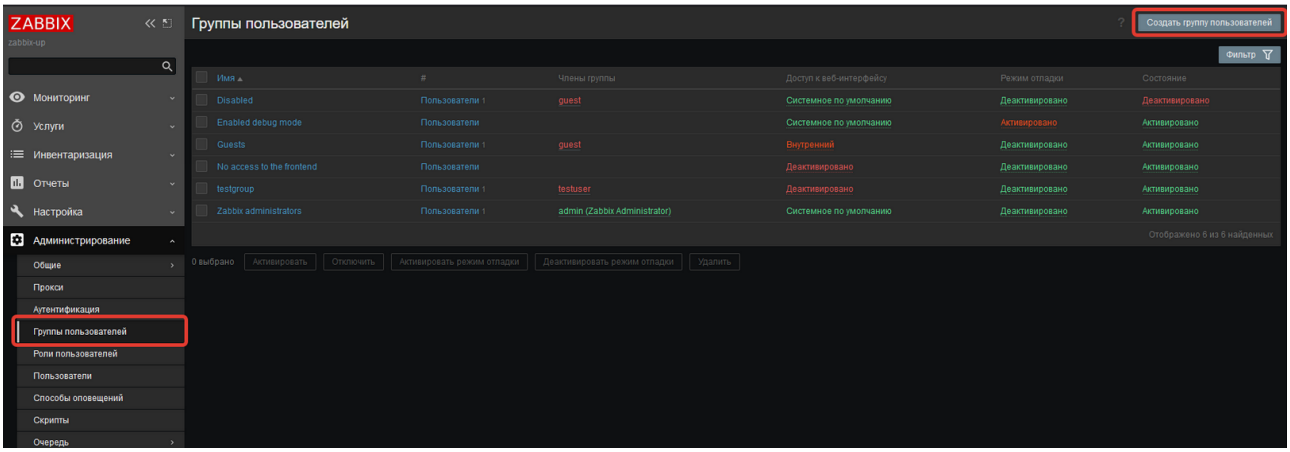
Настройка прав доступа состоит из следующих этапов:

1. Создание группы пользователей.
2. Создание роли.
3. Создание пользователя.

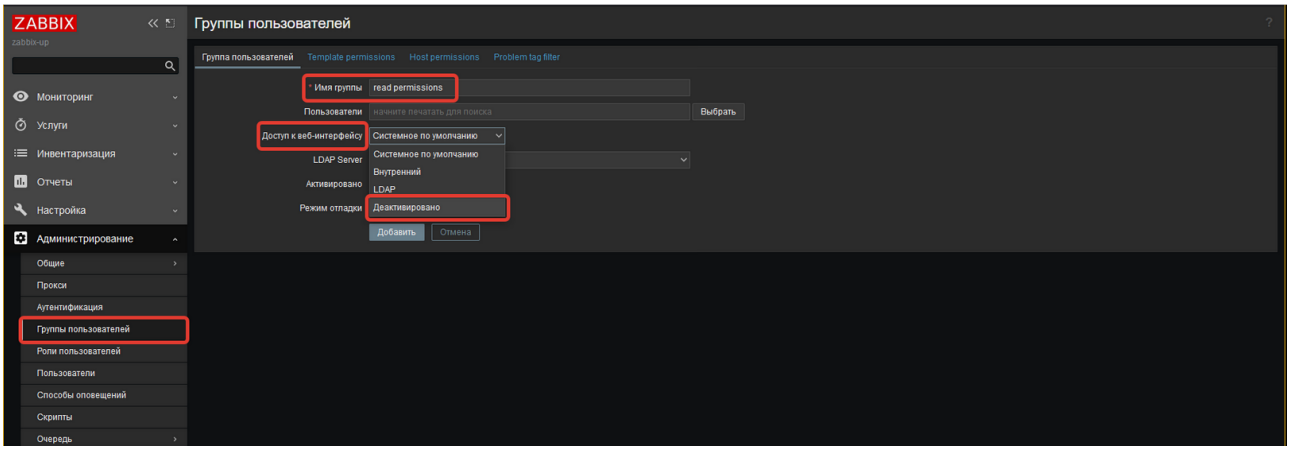
Создание группы пользователей

1. В разделе **Администрирование** выберите пункт **Группы пользователей**.

2. В верхнем правом углу экрана нажмите кнопку **Создать группу пользователей**. Система отобразит окно групп пользователей.

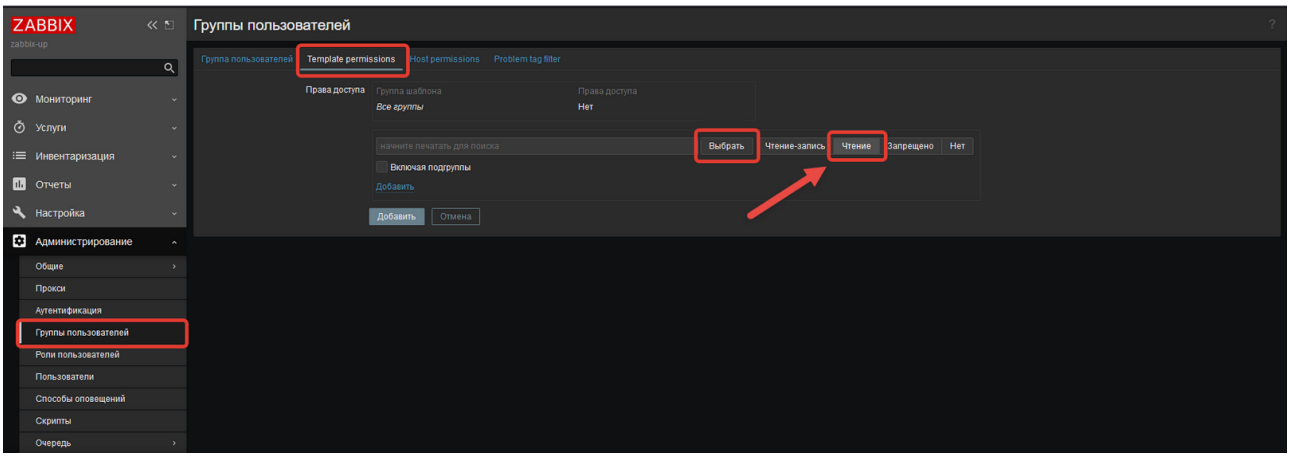


3. Укажите в окне имя группы (в примере — **read permissions**).

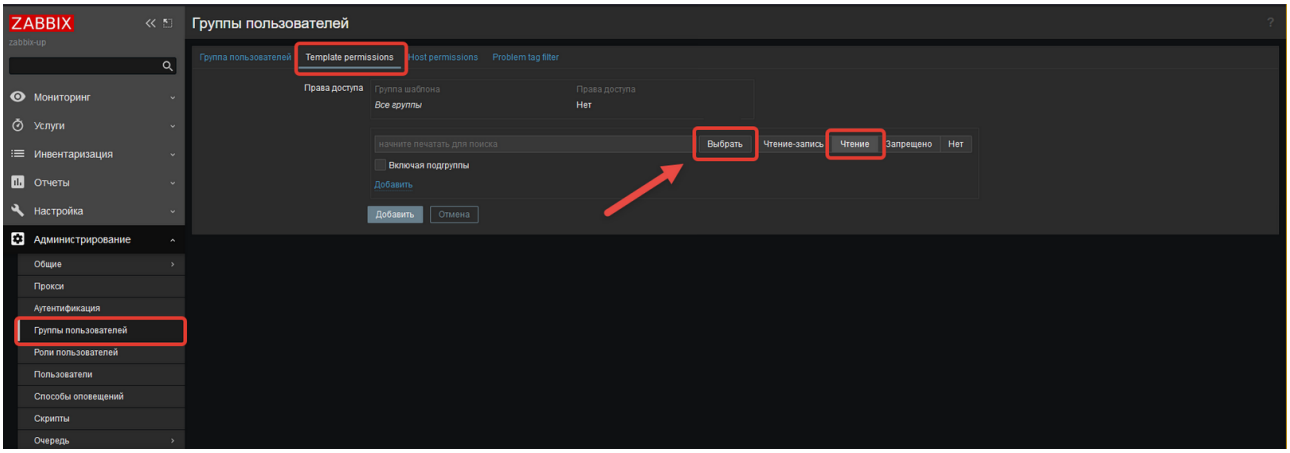


4. В списке **Доступ к интерфейсу** выберите **Деактивировано**.

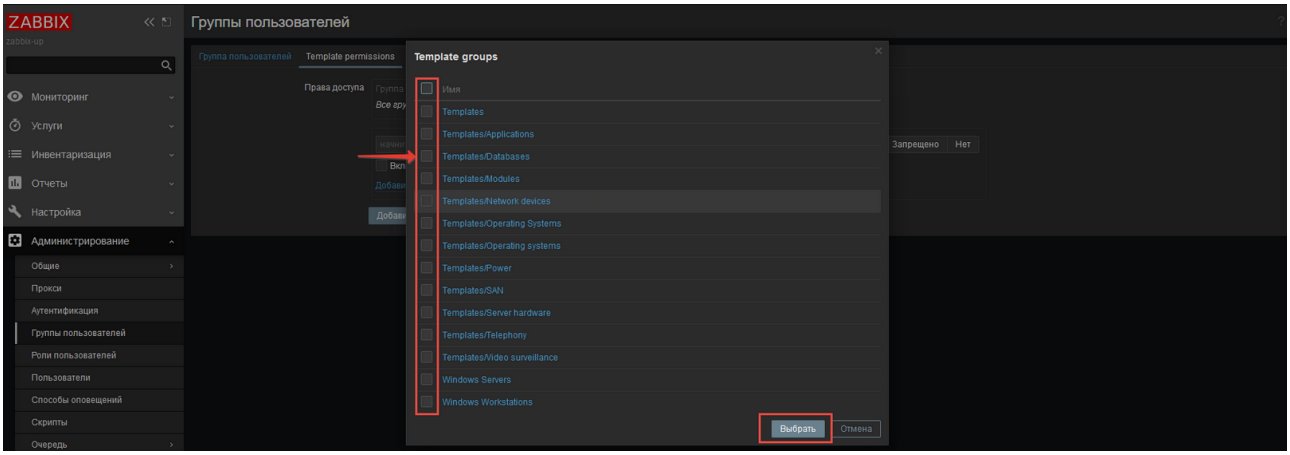
5. Выберите вкладку **Template permissions**, а затем выберите **Чтение**.



6. Нажмите кнопку **Выбрать**.



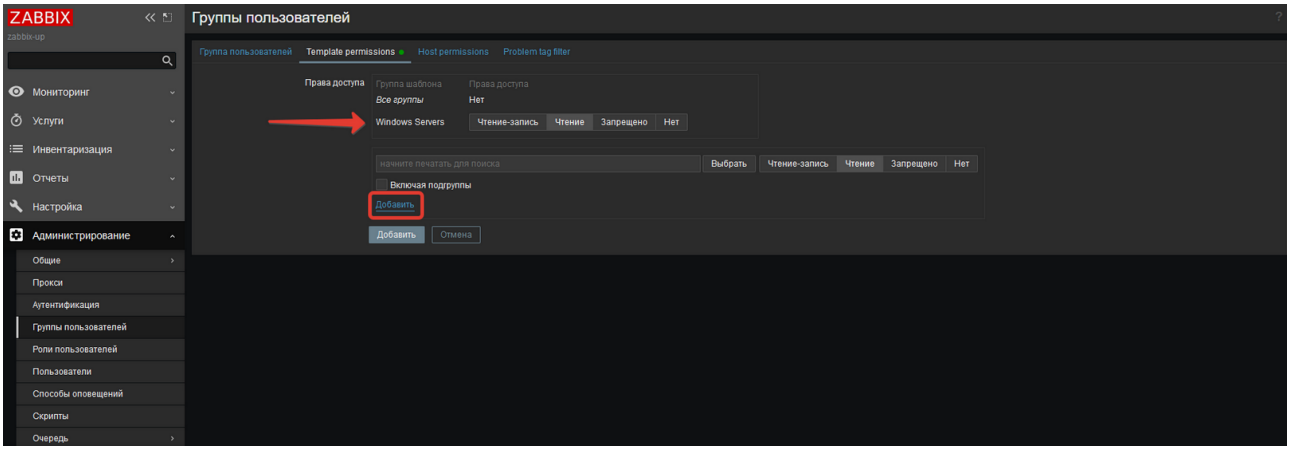
7. В появившемся списке **Template groups** установите флажки для групп, которым необходимо назначить доступ на чтение, а затем нажмите на кнопку **Выбрать** внизу списка.



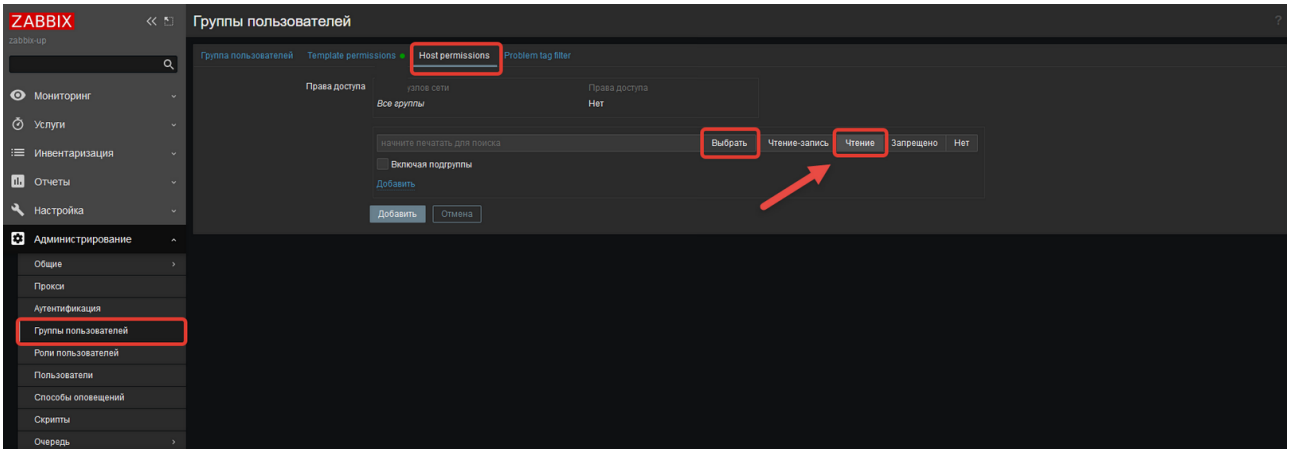
8. Нажмите на **Добавить**.

Обратите внимание, что **Добавить** — это надпись над кнопкой, а не сама кнопка.

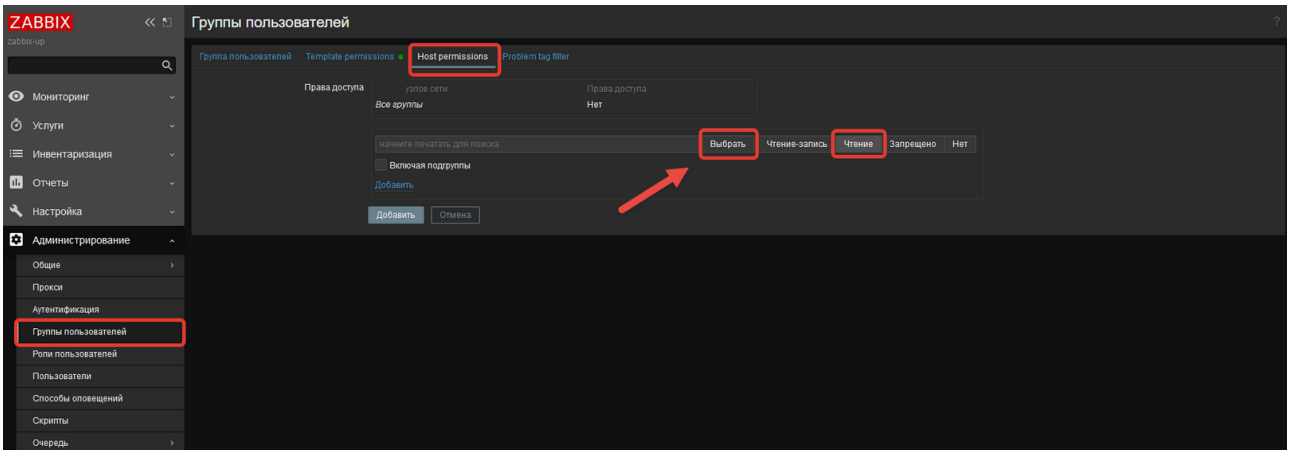
Выбранные группы появятся в списке **Права доступа**.



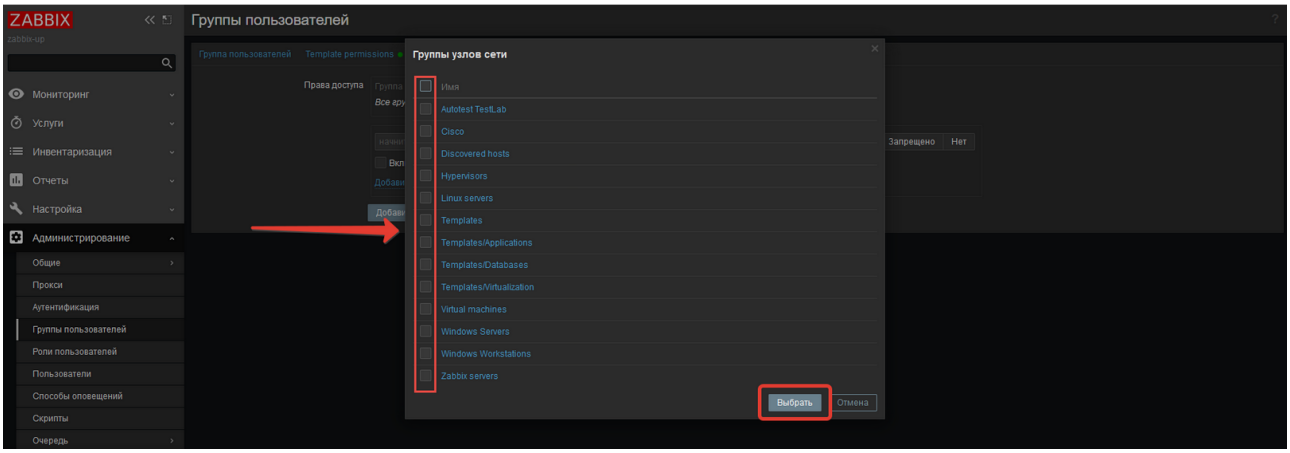
9. Выберите вкладку **Hosts permission**, а затем выберите **Чтение**.



10. Нажмите на кнопку **Выбрать**.



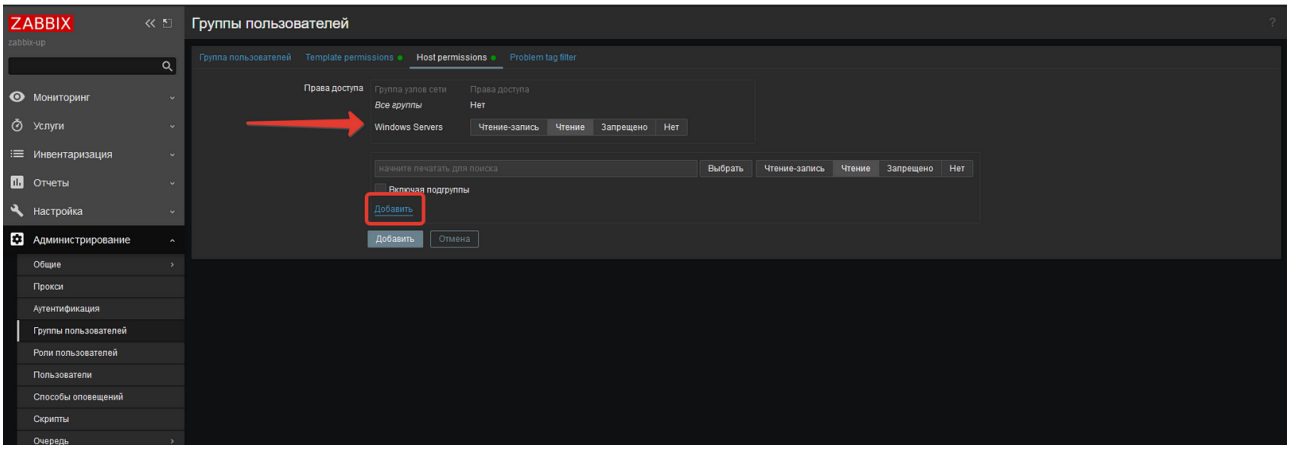
В появившемся списке **Группы узлов сети** установите флажки для групп узлов сети, которым необходимо назначить доступ на чтение, а затем нажмите на кнопку **Выбрать** внизу списка.



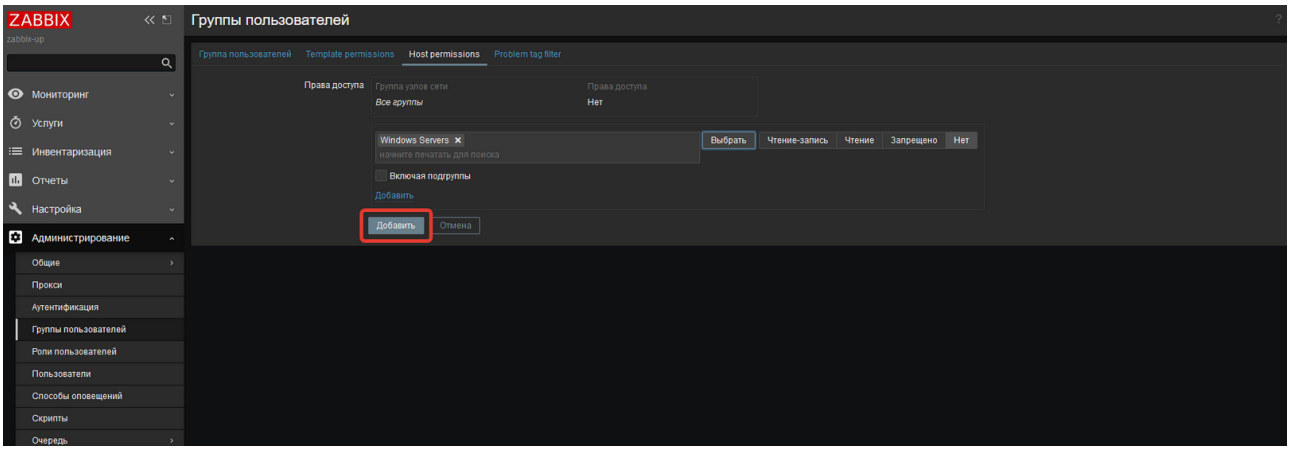
11. Нажмите на **Добавить**.

Обратите внимание, что **Добавить** — это надпись над кнопкой, а не сама кнопка.

Выбранные группы появятся в списке **Права доступа**.

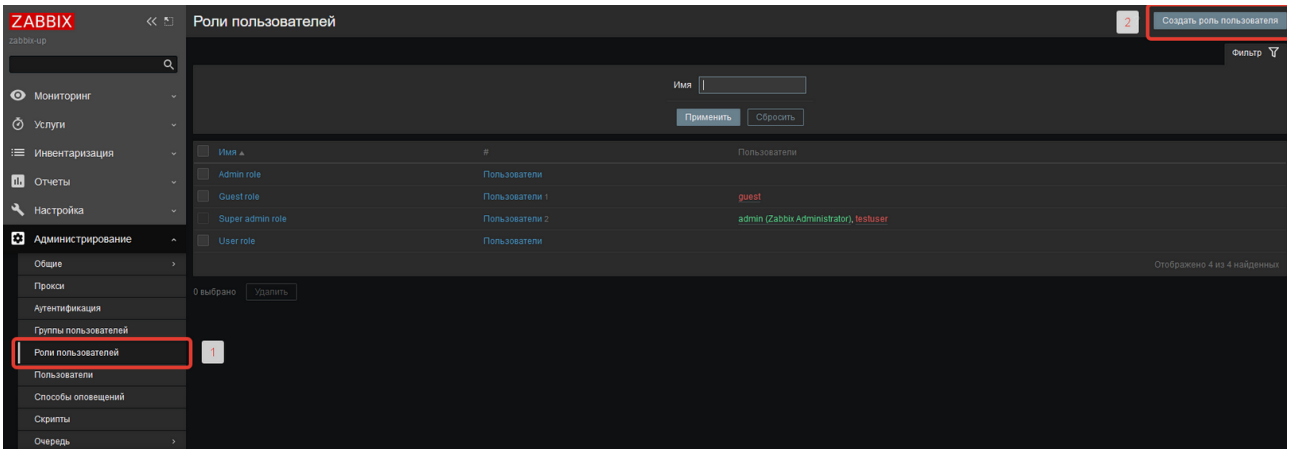


12. Нажмите на кнопку **Добавить**. Система создаст группу.

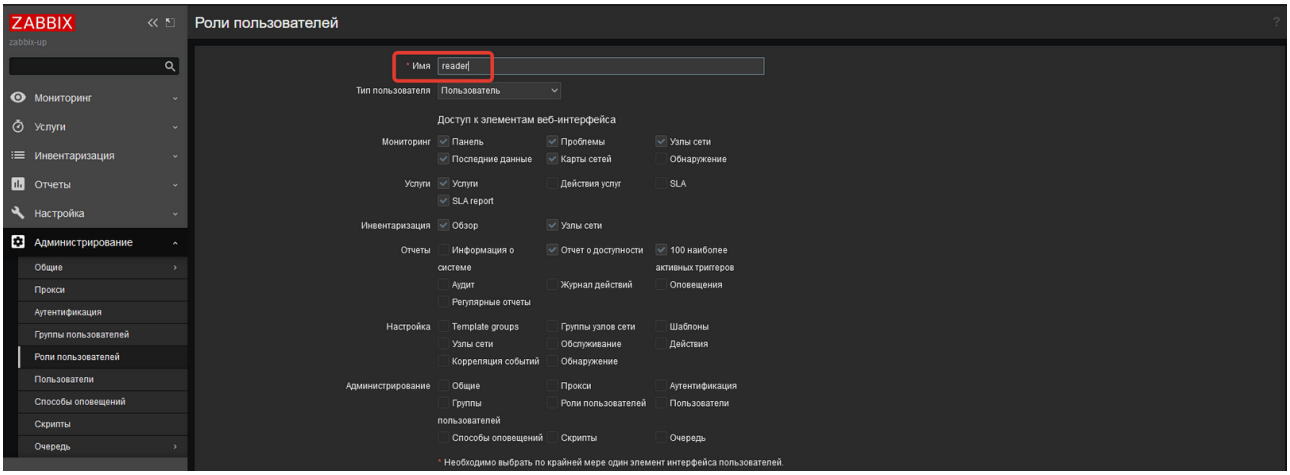


Создание роли

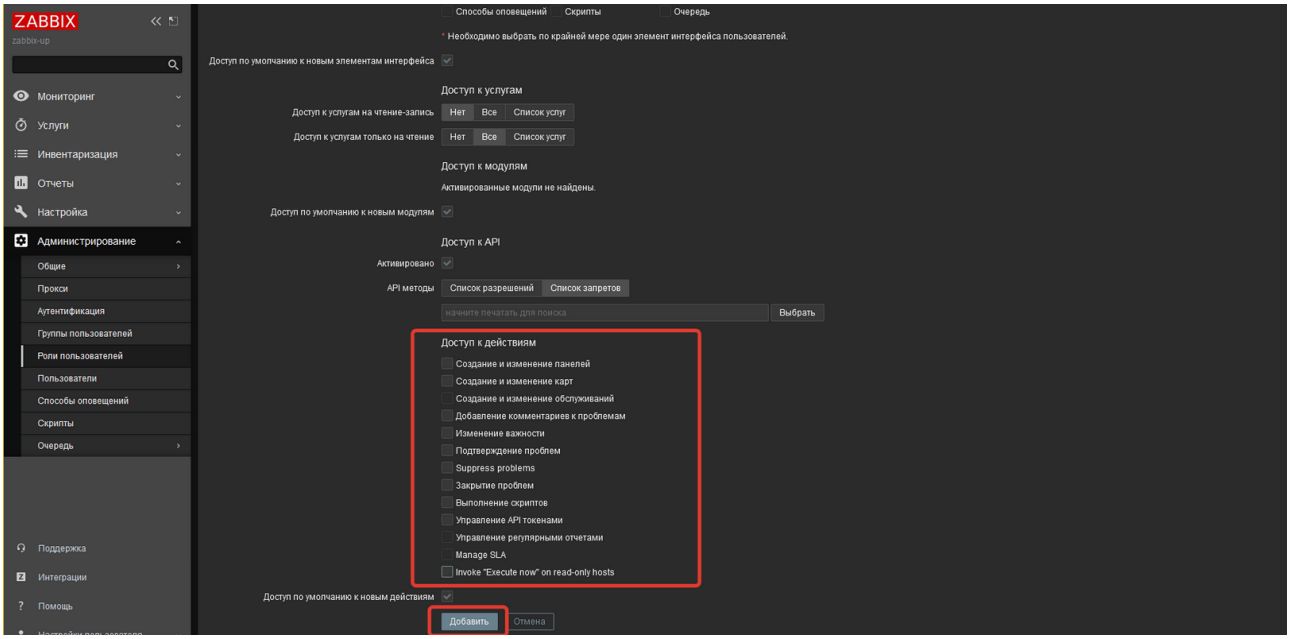
1. В разделе **Администрирование** выберите **Роли пользователей**.
2. В верхнем правом углу экрана нажмите кнопку **Создать роль пользователя**. Система отобразит окно ролей пользователей.



3. Придумайте и укажите имя роли, например, **reader**.

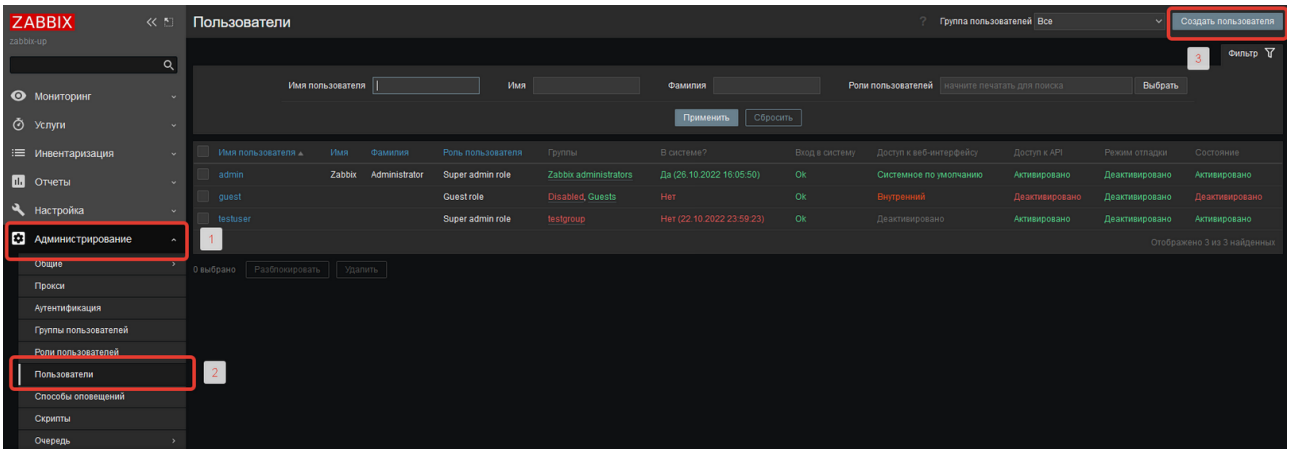


4. Снимите все флажки в разделе **Доступ к действиям**, а затем нажмите на кнопку **Добавить**.

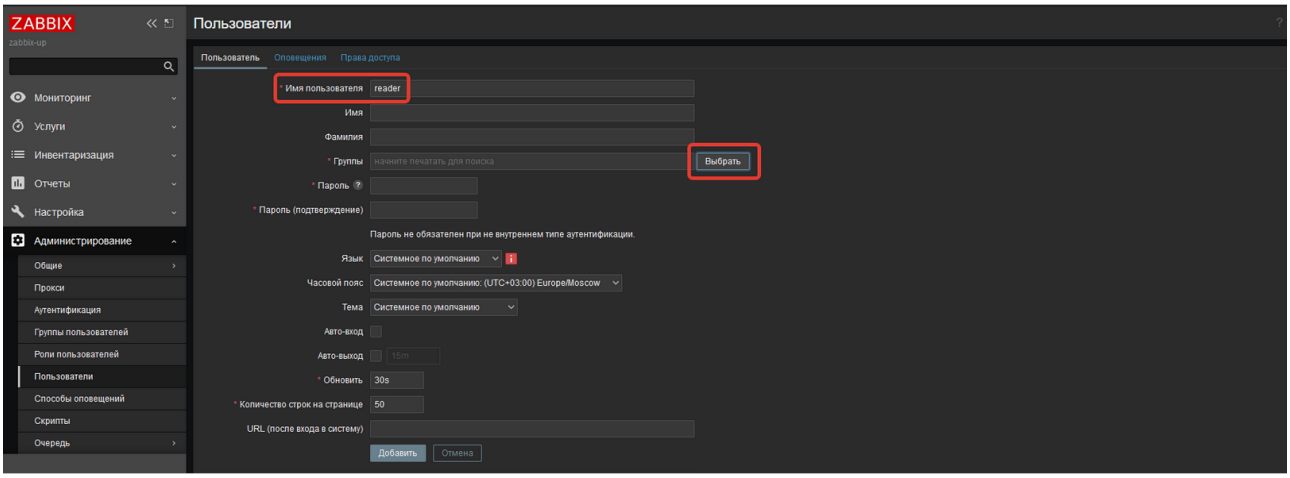


Создание пользователя

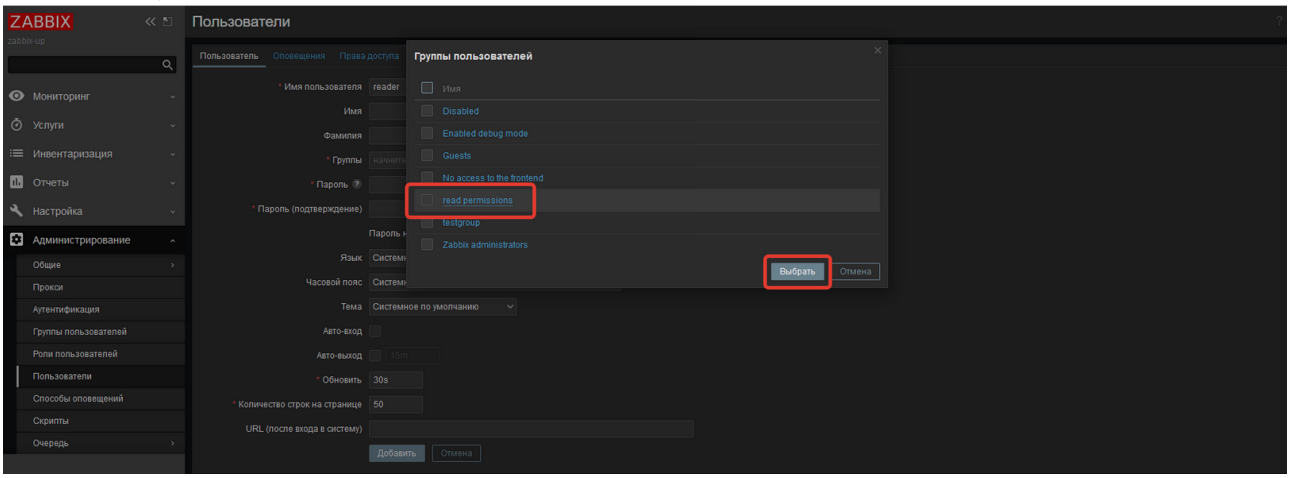
1. В разделе **Администрирование** выберите пункт **Пользователи**.
2. В верхнем правом углу экрана нажмите кнопку **Создать пользователя**. Система отобразит окно пользователей.



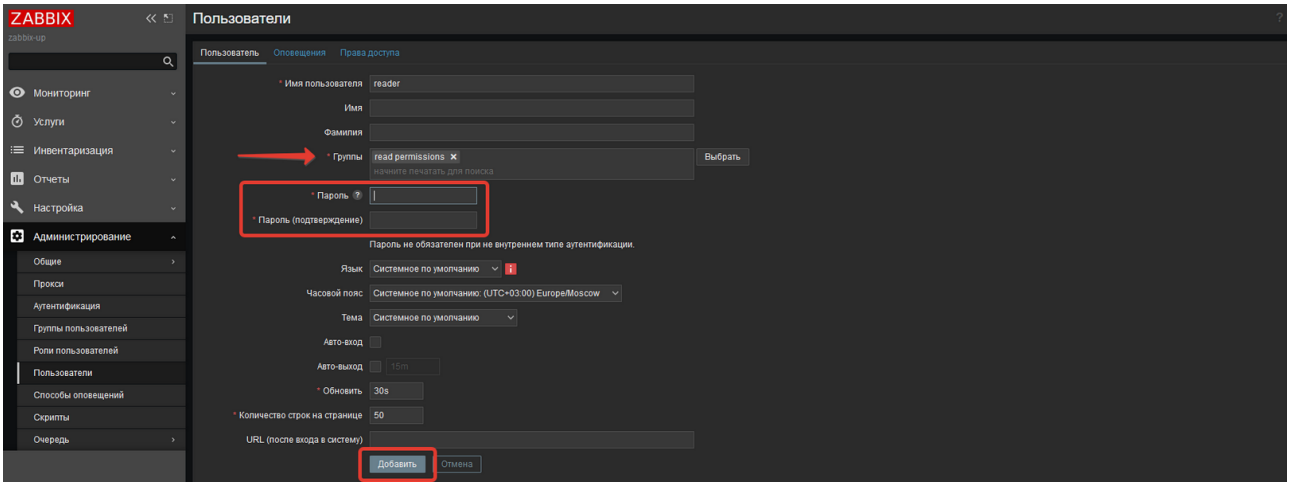
3. Укажите имя пользователя и нажмите на кнопку **Выбрать**.



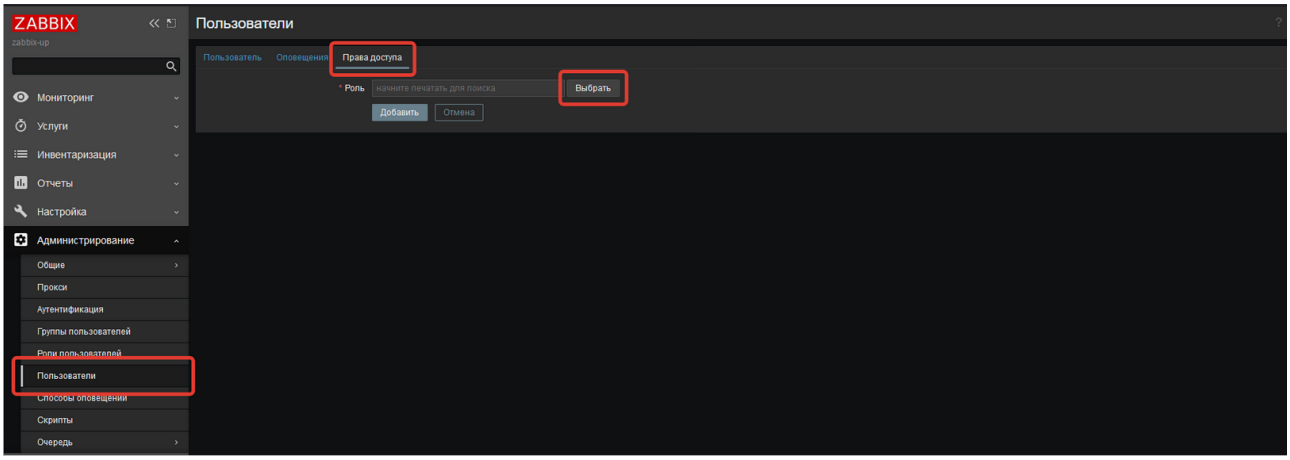
4. Установите в окне флажок рядом с группой, созданной вами ранее (в примере — **read permissions**), а затем нажмите на кнопку **Выбрать**.



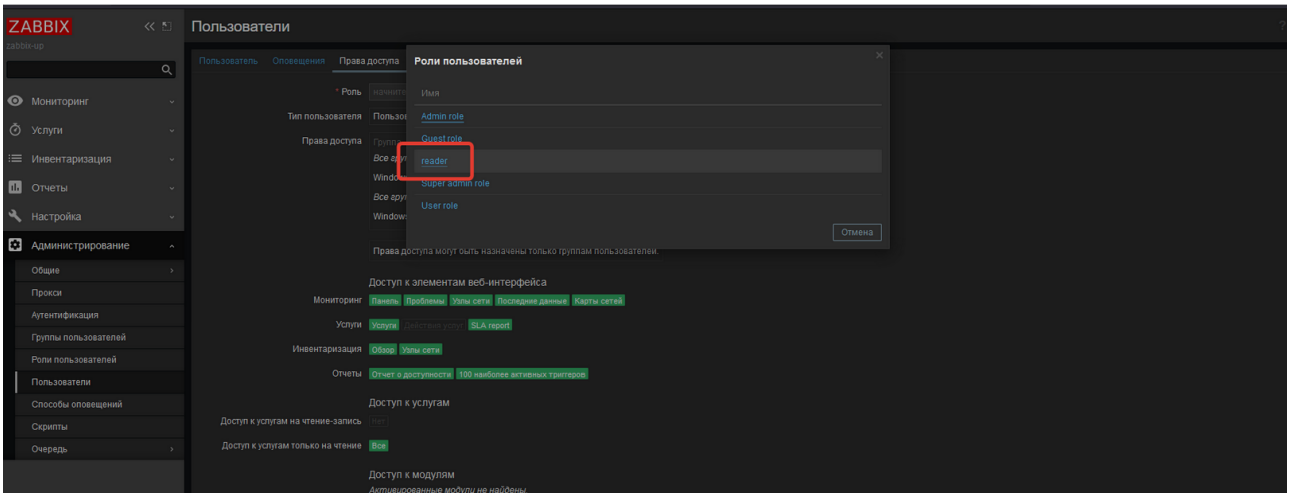
5. Придумайте и укажите пароль, который должен содержать не менее 8 символов. Пароль необходимо подтвердить. Затем нажмите на кнопку **Добавить**.



6. Выберите вкладку **Права доступа** и нажмите на кнопку **Выбрать**.



7. Выберите из списка ранее созданную роль, в примере — **reader**.



8. Нажмите на кнопку **Добавить**. Система создаст пользователя.

Поля в блоке Аргументы

Аргумент	Значение	Редактируемое поле
`Source`	Тип источника данных.	Нет
`ZabbixUrl`	URL-адрес сервера Zabbix. Пример: https://host.domain	Да
`ZabbixLogin`	Логин учетной записи Zabbix.	Да
`ZabbixPassword`	Пароль учетной записи Zabbix.	Да
`ZabbixExtraParameters`	Параметры API-запроса к системе Zabbix. `{ "selectInterfaces": "extend", "selectInventory": "extend" }`	Да

Аргумент	Значение	Редактируемое поле
`EntitiesNormalizations`	Список ID правил нормализации для сущностей, которые необходимо импортировать: • **Оборудование**: zabbix.host.isassets.rvision.ru.Device При необходимости набор импортируемых полей и маппинг можно отредактировать в соответствующем правиле нормализации. Правила нормализации для интеграций располагаются в разделе **Экспертиза** в каталоге **Интеграции**.	Да

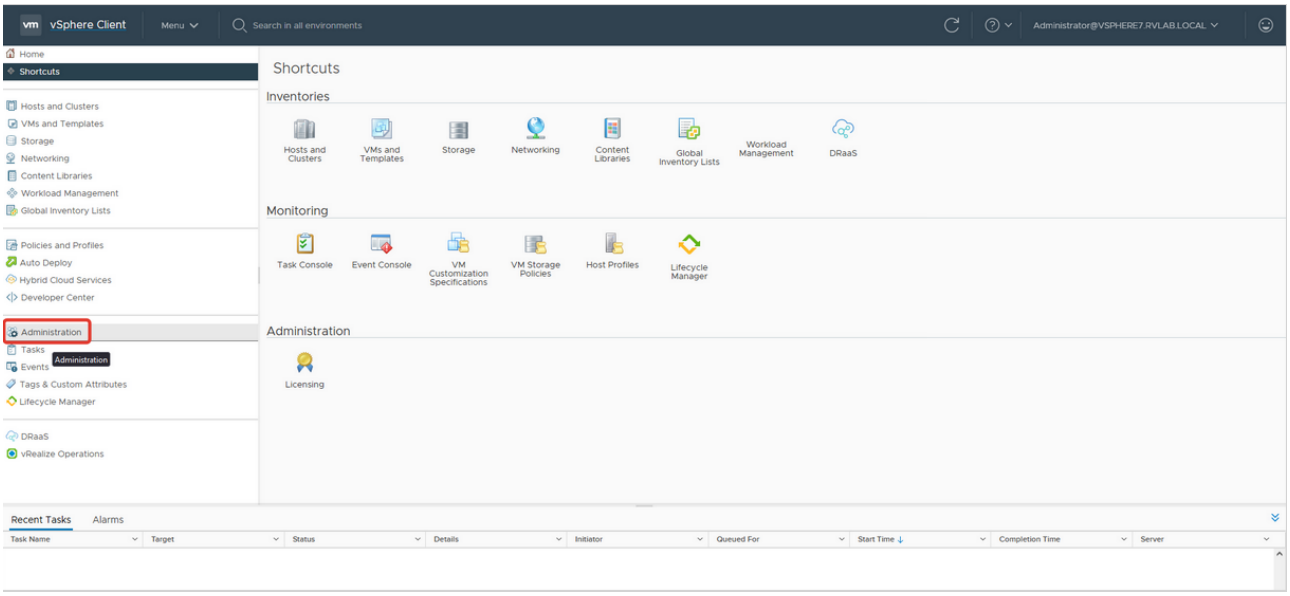
Интеграция с VMware vCenter

При настройке интеграции с VMware vCenter:

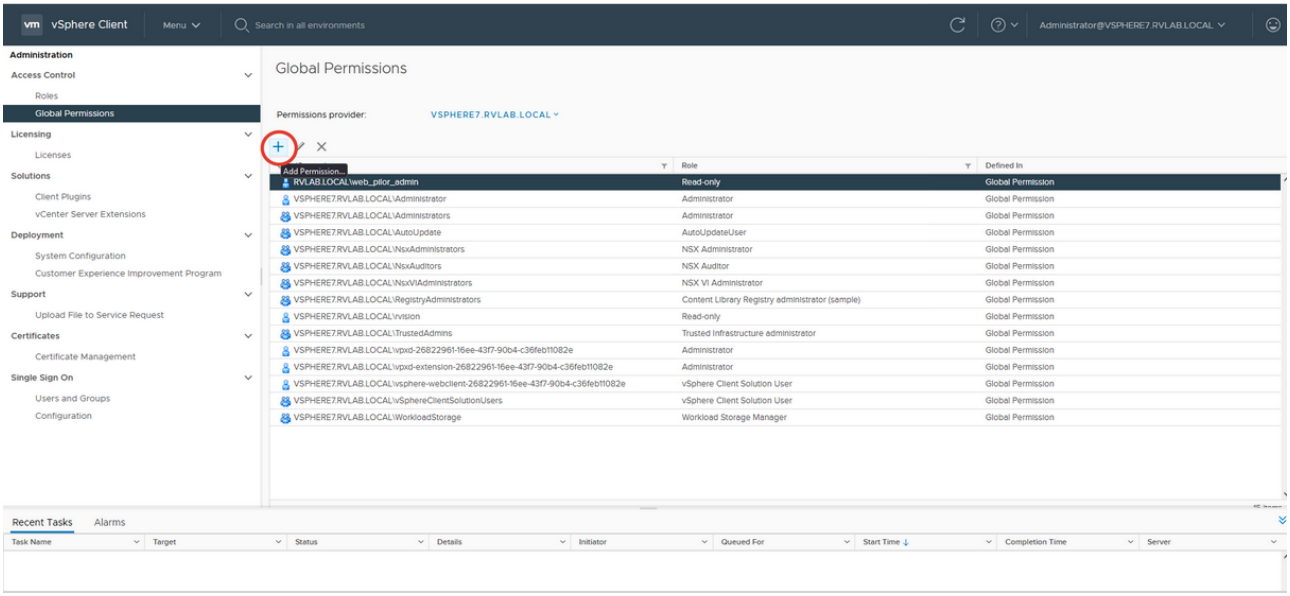
1. Настройте учетную запись.
2. Заполните необходимые поля и активируйте необходимые переключатели в блоке **Аргументы**.

Настройка учетной записи для интеграции с VMware vCenter

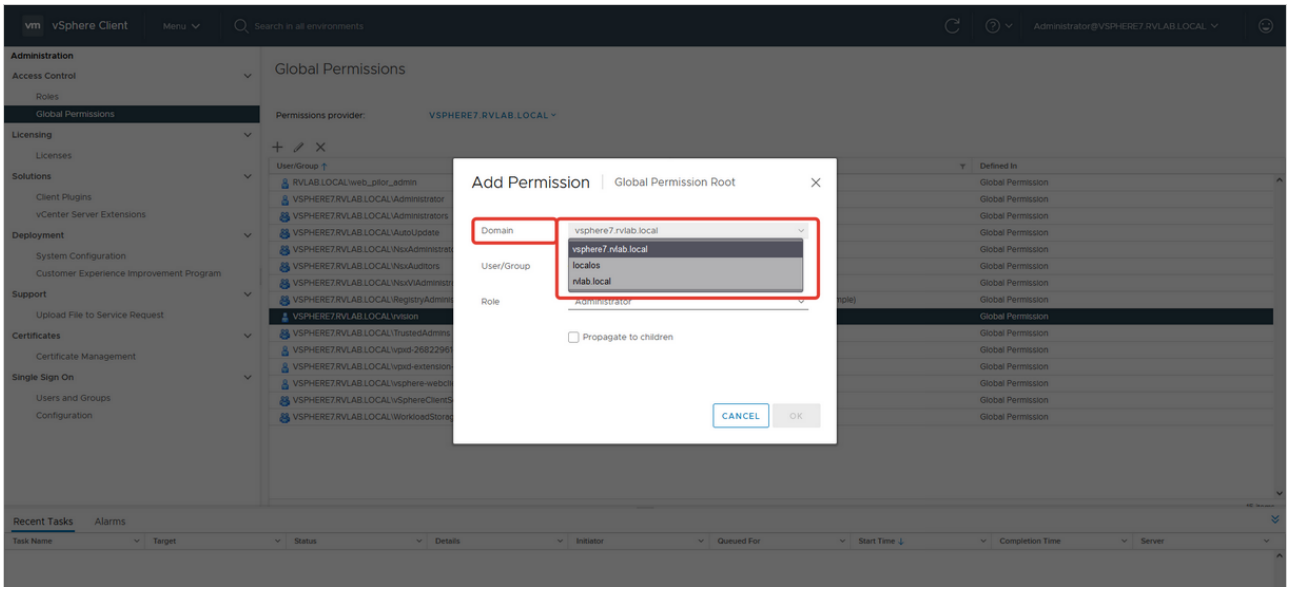
1. Запустите клиент VMware vSphere.
2. Выберите пункт меню **Home** → **Administration**.



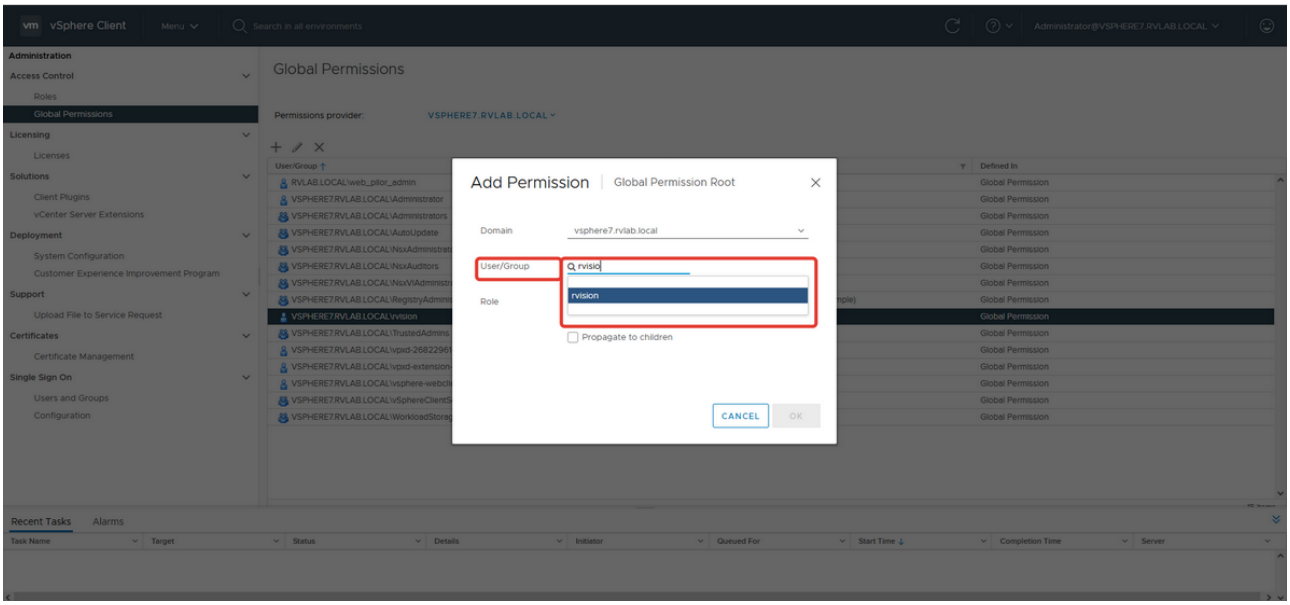
3. В разделе **Global Permission** нажмите кнопку + **(Add Permission)**.



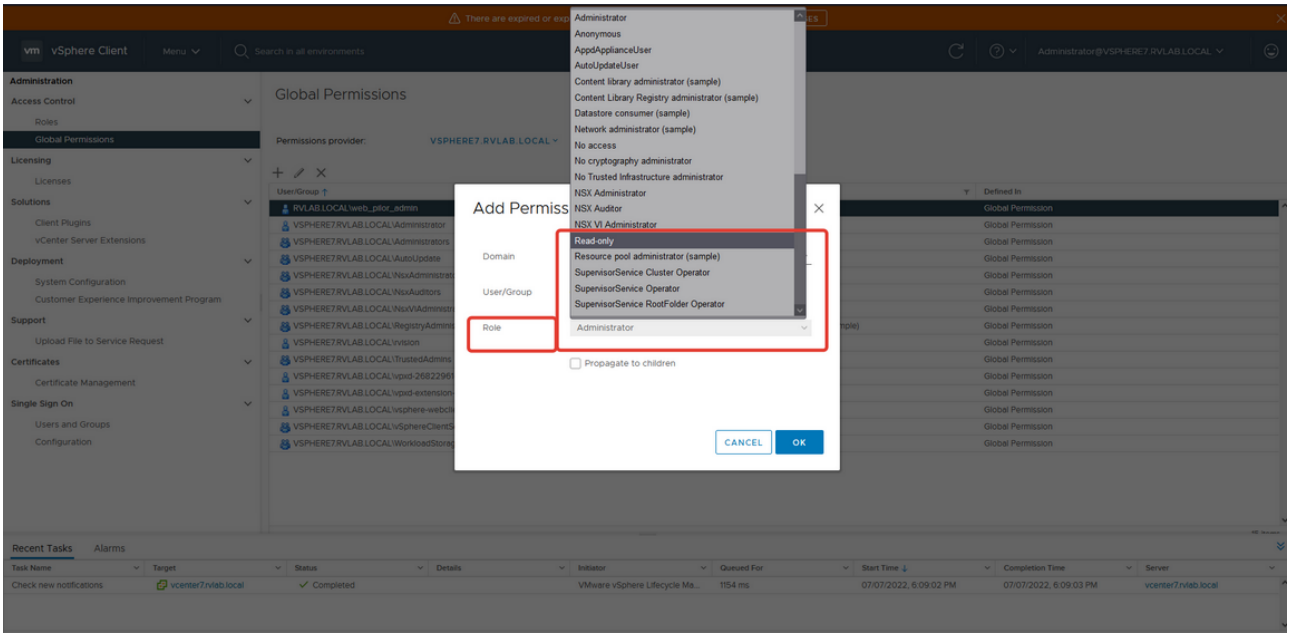
4. В появившемся окне в поле **Domain** выберите домен.



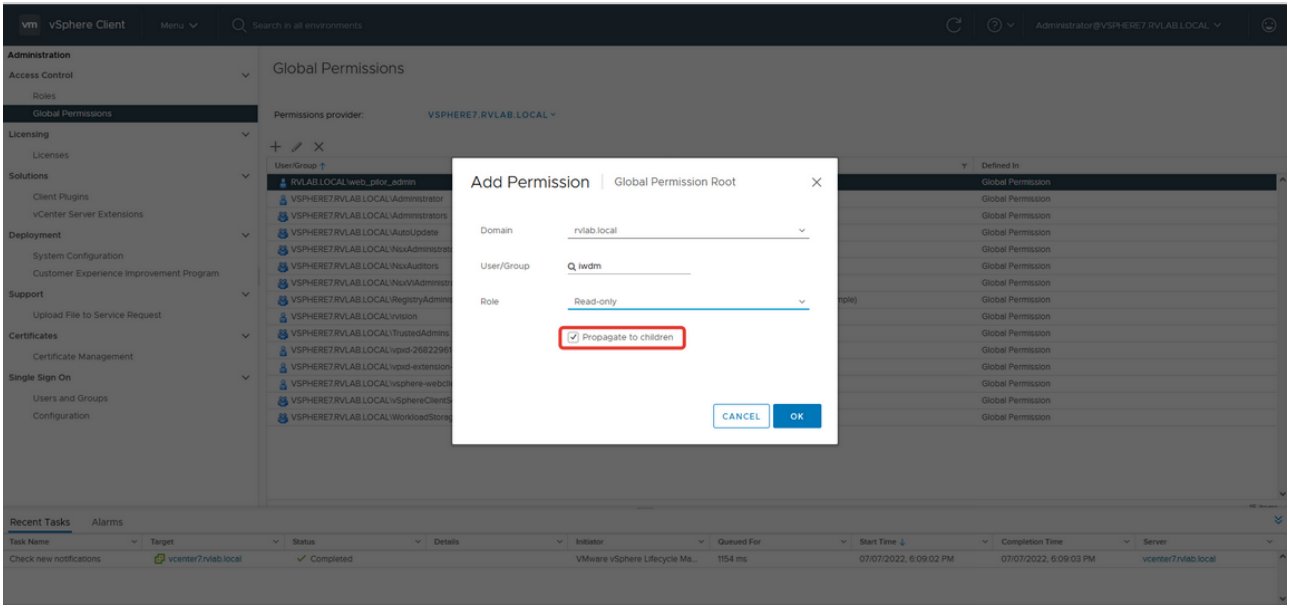
5. В поле **User/Group** выберите пользователя домена, введя несколько первых символов имени.



6. В поле **Role** выберите роль **Read-only**.



7. Убедитесь, что в окне установлен флажок **Propagate to children**.



8. Нажмите **OK**.

Поля в блоке Аргументы

Аргумент	Значение	Редактируемое поле
`LinksNormalizations`	Список ID правил нормализации для связей, которые необходимо импортировать: • Связь **Оборудование — Оборудование**: vmware.isassets.rvision.ru.device_to_device • Связь **Оборудование — Сеть**: vmware.isassets.rvision.ru.network_to_device	Да
`Source`	Тип источника данных.	Нет
`EntitiesNormalizations`	Список ID правил нормализации для сущностей, которые необходимо импортировать: • **Оборудование (виртуальные машины)**: vmware.vm.isassets.rvision.ru.Device • **Сети**: vmware.network.isassets.rvision.ru.Networks • **Оборудование (хосты)**: vmware.host.isassets.rvision.ru.Device При необходимости набор импортируемых полей и маппинг можно отредактировать в соответствующем правиле нормализации. Правила нормализации для интеграций располагаются в разделе **Экспертиза** в каталоге **Интеграции**.	Да
`VMwareLogin`	Логин учетной записи VMWare.	Да

Аргумент	Значение	Редактируемое поле
`VMwarePassword`	Пароль учетной записи VMWare.	Да
`VMwareSSL`	Проверка SSL-сертификатов.	Да
`VMwareUrl`	URL-адрес сервера VMWare Пример: https://host.domain	Да

Переключатели в блоке Аргументы

Аргумент	Значение	Редактируемое поле
`ImportHardware`	Импортировать технические характеристики виртуальной машины.	Да
`ImportHosts`	Импортировать хосты.	Да
`ImportVMNetworkInterfaces`	Импортировать сетевые интерфейсы виртуальной машины.	Да
`ImportVMs`	Импортировать виртуальные машины.	Да
`ImportWithoutVT`	Импортировать гипервизоры и хосты без установленных VMware Tools. Гипервизоры и хосты без установленных VMware Tools будут импортированы без ОС и IP-адреса. Для хостов без VMware Tools имя устройства будет соответствовать названию виртуальной машины в гипервизоре.	Да

Интеграция с Active Directory

При настройке интеграции с Active Directory, заполните необходимые поля и активируйте необходимые переключатели в блоке **Аргументы**.

Поля в блоке Аргументы

Аргумент	Значение	Редактируемое поле
`ADHost`	Адрес сервера AD. Пример: host.domain	Да
`ADPassword`	Пароль учетной записи AD.	Да
`ADSSL`	Проверка SSL-сертификатов.	Да
`ADUsername`	Логин учетной записи AD.	Да
`AccountFilter`	Фильтр для поиска учетных записей. Пример: `name=a368558`	—
`BaseDN`	База поиска в AD.	Да
`ComputerFilter`	Фильтр для поиска оборудования Пример: `distinguishedName=CN=1C83,CN=Computers,DC=rvlab,DC=local`	Да
`GroupFilter`	Фильтр для поиска групп ИТ-активов. Пример: `sAMAccountName=SSOTESTGroup581`	Да

Аргумент	Значение	Редактируемое поле
`LinksNormalizations`	Список ID правил нормализации для связей, которые необходимо импортировать: • Связь **УЗ — Персонал**: active_directory.isassets.rvision.ru.accounts_to_personnel • Связь **Оборудование — Персонал (владелец)**: active_directory.isassets.rvision.ru.device_to_deviceOwner • Связь **Оборудование — ГИТ**: active_directory.isassets.rvision.ru.device_to_assetGroup • Связь **Оборудование — Подразделение**: active_directory.isassets.rvision.ru.device_to_Divisions • связь **Персонал — ГИТ**: active_directory.isassets.rvision.ru.assteGroups_to_Personnel • Связь **Персонал — Подразделение**: active_directory.isassets.rvision.ru.division_to_Personnel	Да
`OrganizationalUnitFilter`	Фильтр для поиска подразделений. Пример: `name=Disabled Computers`	Да
`PagedSize`	Количество записей, возвращаемых за один запрос к AD.	Да
`Source`	Тип источника данных. При необходимости можно указать другое наименование источника, например, если нужно из одной интеграции импортировать данные с разными источниками.	Нет

Аргумент	Значение	Редактируемое поле
`EntitiesNormalizations`	Список ID правил нормализации для сущностей, которые необходимо импортировать: • **Оборудование**: active_directory.computer.isassets.rvision.ru.Device • **Группы ИТ-активов**: active_directory.group.isassets.rvision.ru.AssetGroups • **Учетные записи**: active_directory.user.isassets.rvision.ru.Accounts • **Подразделения**: active_directory.organizationalUnit.isassets.rvision.ru.Divisions • **Персонал**: active_directory.user.isassets.rvision.ru.Personnel При необходимости набор импортируемых полей и маппинг можно отредактировать в соответствующем правиле нормализации. Правила нормализации для интеграций располагаются в разделе **Экспертиза** в каталоге **Интеграции**.	Нет
`UserFilter`	Фильтр для поиска пользователей Пример: `name=a368558`	Да

Переключатели в блоке Аргументы

Аргумент	Значение	Редактируемое поле
`IgnoreLastRun`	Игнорировать дату последнего запуска.	Да
`ImportComputers`	Импортировать оборудование.	Да
`ImportGroups`	Импортировать группы ИТ-активов.	Да
`ImportOrganizationalUnits`	Импортировать подразделения.	Да
`ImportUsers`	Импортировать пользователей.	Да

Настройка расписания параметров интеграции

В блоке **Триггер** вы можете указать параметры триггера типа Расписание или Событие триггера. Запуск будет производиться согласно этим параметрам. Для того чтобы указать параметры, переведите переключатель в верхней части блока **Триггер** в активное положение.

Интеграция с настроенным расписанием отобразится в разделе **Автоматизация** → **Триггеры**. Всю информацию по запускам интеграции можно просмотреть в карточке интеграции на вкладке **Запуски**.

Учетные записи

В этом разделе описывается настройка учетных записей для сканирования и работа с ними.

- Настройка прав учетной записи Windows
  - Конфигурирование службы WinRM вручную
  - Настройка службы WinRM с помощью групповых политик
- Настройка прав учетной записи Linux
- Работа с учетными записями

Настройка прав учетной записи Windows

Для сканирования рабочих станций с помощью сканера уязвимостей используется протокол WinRM — служба удаленного управления для операционных систем Windows. Поддерживается WinRM версии 2.0 и выше.

Убедитесь, что на всех сканируемых рабочих станциях:

- учетная запись для сканирования обладает правами локального администратора;
- служба WinRM включена.

Если имеется только пользовательская учетная запись, ее необходимо настроить согласно описанию в разделах:

- Конфигурирование WinRM вручную

- Конфигурирование WinRM с использованием групповых политик Microsoft Active Directory

Для сбора инвентарной информации и поиска уязвимостей используются следующие команды:

```
# Запросы WMI
Get-WmiObject -Query "SELECT HotFixID,InstalledOn,Description FROM
win32_quickfixengineering"
Get-WmiObject -Query "SELECT FullName,Name,SID,Disabled,LocalAccount FROM
Win32_UserAccount Where LocalAccount = True"
Get-WmiObject -Query "SELECT Description,Index,IPAddress,IPSubnet,MACAddress,MTU FROM
Win32_NetworkAdapterConfiguration WHERE MACAddress != NULL"
Get-WmiObject -Query "SELECT Caption FROM Win32_OperatingSystem"
Get-WmiObject -Query "SELECT \* FROM Win32_ComputerSystem"

# Запросы к реестру
REG QUERY "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall" /s
REG QUERY "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion" /v
ProductName
REG QUERY "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\\CurrentVersion" /v UBR
```

Конфигурирование службы WinRM вручную

Настройка службы WinRM

В командной строке с повышенными привилегиями запустите команду `winrm qc`. Команда выполняет следующие действия:

- Устанавливает режим запуска службы WinRM в **Automatic (Delayed Start)**.
- Запускает службу WinRM.
- Создает правила для Firewall. WinRM работает на порту 5985 по протоколу HTTP.

▼ *Пример вывода команды* `winrm qc`

```
WinRM is not set up to receive requests on this machine.
The following changes must be made:

Start the WinRM service.

Make these changes [y/n]? y

WinRM has been updated to receive requests.

WinRM service started.
WinRM already is set up for remote management on this machine.
```

Если WinRM сконфигурирован, можно проверить конфигурацию listener следующей командой:

```
winrm enumerate winrm/config/listener
```

▼ *Пример вывода команды* `winrm enumerate winrm/config/listener`

```
Listener
    Address = \*
    Transport = HTTP
    Port = 5985
    Hostname
    Enabled = true
    URLPrefix = wsman
    CertificateThumbprint
    ListeningOn = 10.99.101.87, 127.0.0.1, ::1, fe80::5efe:10.99.101.87%12,
fe80::ad59:dfe3:f2:bdd0%11
```

Просмотреть всю конфигурацию можно следующей командой:

```
winrm get winrm/config
```

▼ *Пример вывода команды* `winrm get winrm/config`

```
Config
    MaxEnvelopeSizekb = 150
    MaxTimeoutms = 60000
    MaxBatchItems = 32000
    MaxProviderRequests = 4294967295
    Client
        NetworkDelayms = 5000
        URLPrefix = wsman
        AllowUnencrypted = false
        Auth
            Basic = true
            Digest = true
            Kerberos = true
            Negotiate = true
            Certificate = true
            CredSSP = false
        DefaultPorts
            HTTP = 5985
            HTTPS = 5986
        TrustedHosts
    Service
        RootSDDL = O:NSG:BAD:P(A;;GA;;;BA)(A;;GA;;;S-1-5-21-666545053-1111427937
-1989007194-1003)S:P(AU;FA;GA;;;WD)(AU;SA;GXGW;;;WD)
        MaxConcurrentOperations = 4294967295
        MaxConcurrentOperationsPerUser = 15
        EnumerationTimeoutms = 60000
        MaxConnections = 25
        MaxPacketRetrievalTimeSeconds = 120
        AllowUnencrypted = false
        Auth
```

```
            Basic = false
            Kerberos = true
            Negotiate = true
            Certificate = false
            CredSSP = false
            CbtHardeningLevel = Relaxed
        DefaultPorts
            HTTP = 5985
            HTTPS = 5986
        IPv4Filter = \*
        IPv6Filter = \*
        EnableCompatibilityHttpListener = false
        EnableCompatibilityHttpsListener = false
        CertificateThumbprint
    Winrs
        AllowRemoteShellAccess = true
        IdleTimeout = 180000
        MaxConcurrentUsers = 5
        MaxShellRunTime = 2147483647
        MaxProcessesPerShell = 15
        MaxMemoryPerShellMB = 150
        MaxShellsPerUser = 5
```

Настройка доступа для учетных записей без прав администратора

Чтобы настроить доступ для учетных записей без прав администратора, нужно создать доменную или локальную учетную запись, от имени которой будет осуществляться инвентаризация оборудования.

Права для дескриптора безопасности WinRM

Чтобы настроить права для дескриптора безопасности WinRM:

1. В командной строке с повышенными привилегиями запустите следующую команду:

   ```
   winrm configsddl default
   ```

2. Добавьте учетную запись с правами **Full Control**, от имени которой будет осуществляться инвентаризация оборудования.

Добавление прав на доступ к пространствам имен WMI (root/CIMV2, root/SECURITY)

Чтобы добавить права на доступ к пространствам имен:

1. В командной строке с повышенными привилегиями запустите оснастку **Элементы управления WMI**.
2. Откройте окно **Свойства** элемента управления WMI.
3. Перейдите на вкладку **Безопасность**.
4. Выделите ветку **Root** и нажмите на кнопку **Безопасность**.

5. Откройте окно **Дополнительные параметры безопасности.**
6. Добавьте учетную запись пользователя для инвентаризации оборудования.
7. Настройте права учетной записи: установите флажки **Разрешить** для:
   - **Выполнение методов** (ExecuteMetods).
   - **Включить учетную запись** (EnableAccount).
   - **Включить удаленно** (RemoteEnable).
8. В списке **Применять** укажите область применения: **Данное пространство и подпространство имен** (This namespace and subnamespaces).
9. Чтобы сохранить изменения, нажмите на кнопку **ОК**.

Настройка службы WinRM с помощью групповых политик

Вы можете централизованно настроить службу WinRM с помощью групповых политик домена Windows.

В руководстве приведен пример настройки групповых политик домена Windows. Настройки могут отличаться в зависимости от особенностей ИТ-инфраструктуры и положений внутренних нормативных документов организации.

Добавление исключений для брандмауэра

Чтобы добавить исключения для брандмауэера:

1. Перейдите в раздел **Конфигурация Компьютера** → **Политики** → **Конфигурация Windows** → **Параметры безопасности** → **Монитор брандмауэра Защитника Windows в режиме повышенной безопасности** → **Правила для входящих подключений**.
2. Создайте правило для нового входящего подключения. В мастере создания правила выберите тип правила **Предопределенные** → **Удаленное управление Windows**.
3. Нажмите на кнопку **Далее**.
4. В списке предопределенных правил выберите **Удаленное управление Windows (HTTP - входящий трафик)**.
5. Нажмите на кнопку **Далее**.
6. На экране выбора действий выберите **Разрешить подключение**.
7. Нажмите на кнопку **Готово**. Настройка завершена.

Настройка автоматического запуска службы WinRM

При настройке используйте консоль Group Policy Management Console (GPMC.msc).

Чтобы настроить автоматический запуск службы WinRM:

1. Перейдите в раздел **Конфигурация Компьютера** → **Настройка** → **Службы**.
2. Выберите из списка службу WinRM. Если она отсутствует, добавьте новую службу со следующими параметрами:

a. Вкладка **Общие**:

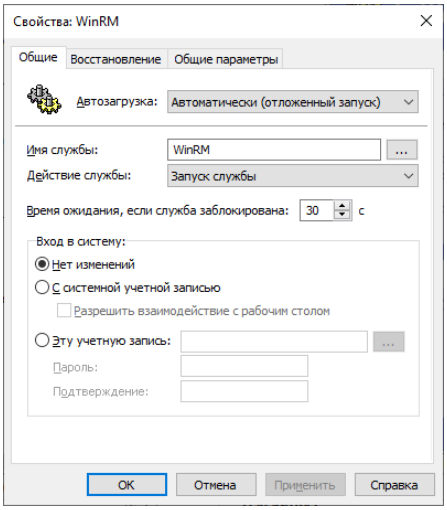
   i. Имя службы: WinRM.

   ii. Автозагрузка: Автоматически.

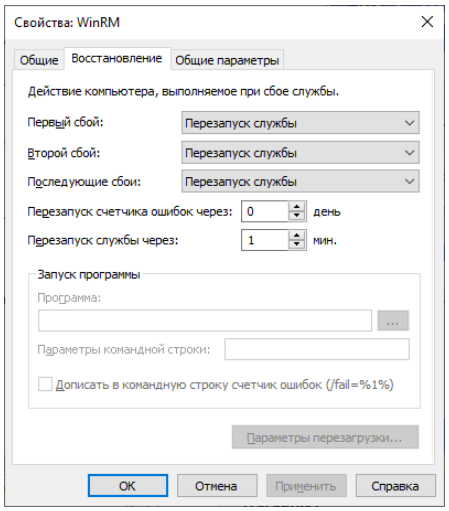
   iii. Действие службы: Запуск службы.

   iv. Время ожидания, если служба заблокирована: 30 с.

   v. Вход в систему: Нет изменений.



b. Вкладка **Восстановление**: Действие компьютера, выполняемое при сбое службы - Перезапуск службы (во всех вложенных меню — Первый сбой, Второй сбой, Последующие сбои).



3. Нажмите на кнопку **ОК**. Служба будет добавлена в систему.

Настройка адреса для удаленного управления

Чтобы настроить адрес для удаленного управления:

1. Перейдите в раздел **Конфигурация Компьютера → Политики → Административные шаблоны → Компоненты Windows → Удаленное управление Windows → Служба удаленного управления Windows.**
2. Выберите политику **Разрешить удаленное администрирование сервера средствами WinRM**.
3. Установите статус политики **Включено**.

4. В окне настроек политики добавьте перечень IP-адресов коллекторов R-Vision, которые осуществляют сканирование оборудования.
5. Нажмите на кнопку **ОК**. Настройка завершена.

Добавление прав на доступ к пространствам имен WMI (root/CIMV2, root/SECURITY) и на дескриптор безопасности WMI

Чтобы настроить права:

1. Перейдите в раздел **Конфигурация Компьютера** → **Политики** → **Конфигурация Windows** → **Сценарии (запуск/завершение)**.
2. Откройте сценарий **Автозагрузка**.
3. Добавьте в сценарий автозагрузки два скрипта:
   a. Скрипт, предоставляющий доступ пользователю к пространствам имен WMI root/CIMV2, root/SECURITY.

      Имя сценария: `Set-WmiNamespaceSecurity.ps1`. Чтобы получить файл скрипта, обратитесь в службу поддержки по адресу support@rvision.com.

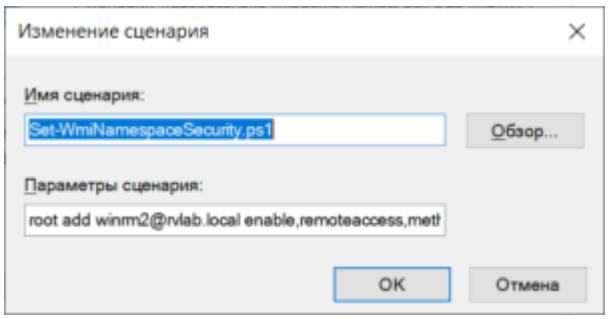
      Параметры:

      ```
      <wmi-namespace> add <username> enable,remoteaccess,methodexecute
      ```

      Здесь:

      - `<wmi-namespace>` — пространство имен WMI;
      - `<username>` — пользователь, от имени которого осуществляется инвентаризация.

        Для корректной инвентаризации, пользователю требуется доступ к пространствам WMI: либо к root/CIMV2 и root/SECURITY, либо к root.

   b. Скрипт, предоставляющий права доступа пользователю на дескриптор безопасности WMI.

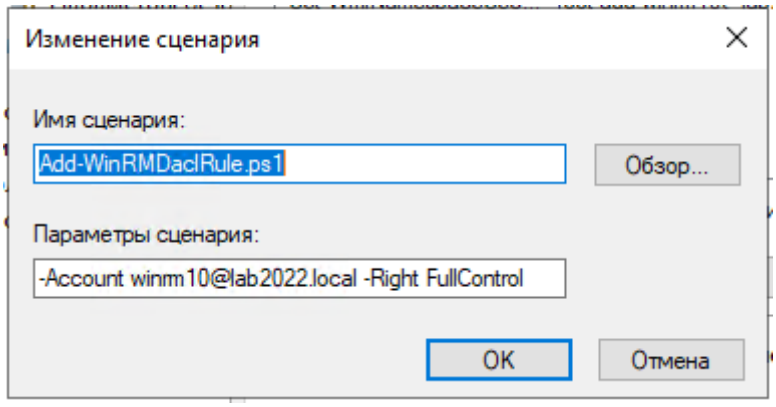
      Имя сценария: `Add-WinRMDaclRule.ps1`. Чтобы получить файл скрипта, обратитесь в службу поддержки по адресу support@rvision.ru.

      Параметры:

      ```
      Account <username> -Right FullControl
      ```

Здесь:

- `<username>` — пользователь, от имени которого осуществляется инвентаризация.



Настройка прав учетной записи Linux

Требования к учетной записи для сканирования

Для сканирования рабочих станций с помощью сканера уязвимостей требуется учетная запись с авторизацией по паролю и правами на выполнение следующих команд:

- `uname`
- `ls`
- `hostname`
- `dnsdomainname`
- `cat`
- `ip addr`
- `rpm -qa`
- `dpkg-query -W`
- `getent passwd`
- `lastlog`
- `lsb_release`
- `hostnamectl`
- `grep`

Требования к доступности файлов для сканирования

Ниже приведен список файлов, которые должны быть доступны для чтения командой `cat`. Как правило, они доступны для пользователей по умолчанию, но, возможно, потребуется дополнительная настройка доступа.

- Получение сведений об установленной ОС:
  - `/etc/altlinux-release`
  - `/etc/centos-release`

    - /etc/oracle-release
    - /etc/redhat-release
    - /etc/lsb-release
    - /etc/os-release
- Поиск уязвимостей:
    - /boot/grub2/grubenv
    - /boot/grub2/grub.cfg
    - /etc/dnf/modules.d/ruby.module
    - /etc/dnf/modules.d/pki-deps.module
    - /etc/dnf/modules.d/pki-core.module
    - /etc/dnf/modules.d/idm.module
    - /etc/dnf/modules.d/container-tools.module
    - /etc/dnf/modules.d/nodejs.module
    - /etc/dnf/modules.d/httpd.module
    - /etc/dnf/modules.d/python27.module
    - /etc/dnf/modules.d/python36.module
    - /etc/dnf/modules.d/freeradius.module
    - /etc/dnf/modules.d/virt.module
    - /etc/dnf/modules.d/go-toolset.module
    - /etc/dnf/modules.d/redis.module
    - /etc/dnf/modules.d/mysql.module
    - /etc/dnf/modules.d/subversion.module
    - /etc/dnf/modules.d/squid.module
    - /etc/dnf/modules.d/nginx.module
    - /etc/dnf/modules.d/virt-devel.module
    - /etc/dnf/modules.d/389-ds.module
    - /etc/dnf/modules.d/mariadb.module
    - /etc/dnf/modules.d/mariadb-devel.module
    - /etc/dnf/modules.d/php.module
    - /etc/dnf/modules.d/mod_auth_openidc.module
    - /etc/dnf/modules.d/postgresql.module
    - /etc/dnf/modules.d/python38.module
    - /etc/dnf/modules.d/mailman.module
    - /etc/dnf/modules.d/varnish.module
    - /etc/dnf/modules.d/rust-toolset.module

- `/etc/dnf/modules.d/python38-devel.module`
- `/etc/dnf/modules.d/python39.module`
- `/etc/dnf/modules.d/python39-devel.module`
- `/etc/dnf/modules.d/llvm-toolset.module`
- `/etc/dnf/modules.d/parfait.module`
- `/etc/dnf/modules.d/maven.module`
- `/proc/modules`
- `/etc/debian_version`
- `/snap/bin/canonical-livepatch`

Работа с учетными записями

Работа с учетными записями осуществляется в разделе **Сканирование** → **Учетные записи**.

Об учетных записях

Учетная запись содержит данные для аутентификации на оборудовании в процессе его инвентаризации и поиска уязвимостей. Учетная запись создается для организации, поэтому в задаче для определенной организации будут доступны только учетные записи этой организации.

Интерфейс раздела

Панель инструментов включает в себя следующие компоненты:

- Кнопка **Добавить** (＋) позволяет добавить новую учетную запись.
- Поле **Поиск** предназначено для быстрого поиска учетных записей в списке по полю **Название**.
- Выпадающий список для выбора тенанта позволяет отфильтровать список учетных записей по тенанту.

В рабочей области отображается таблица имеющихся учетных записей, которая содержит следующие столбцы:

- **Тип** — тип учетной записи. Доступные типы:
  - SSH;
  - Windows;
  - СУБД.
- **Название** — название учетной записи.
- **Логин** — логин учетной записи.
- **Статус** — статус учетной записи: **Включена** или **Отключена**.
- **Тенант** — тенант, к которому относится учетная запись.
- **Дата обновления** — дата последнего обновления учетной записи.
- **Дата создания** — дата создания учетной записи.

При работе с таблицей учетных записей доступны следующие операции:

- Поиск учетных записей по полю **Название**.
- Фильтрация учетных записей по полю **Тенант**.
- Сортировка учетных записей по полям **Тип**, **Название**, **Логин**, **Статус**, **Дата создания**, **Дата обновления**.

При выборе учетной записи в правой части экрана отображается ее карточка с детальной информацией.

Работа с учетными записями

Доступные операции над учетными записями:

- Создание учетной записи
- Просмотр учетной записи
- Изменение учетной записи
- Удаление учетной записи
- Проверка учетной записи

Создание учетной записи

Вы можете создать учетную запись следующих типов:

- SSH;
- Windows;
- СУБД.

Создание учетной записи типа SSH

Чтобы создать учетную запись типа **SSH**:

1. Нажмите на кнопку **Добавить** (+). Система отобразит выпадающий список возможных типов учетной записи.
2. Выберите тип **SSH**. Система отобразит окно добавления учетной записи Linux.
3. Выберите в выпадающем списке тенант, в котором будет создана учетная запись.
4. Укажите название и описание учетной записи.
5. В разделе **Параметры аутентификации** выберите способ аутентификации:
   - **Пароль**: введите логин и пароль, используемые для аутентификации.
   - **Публичный ключ**:
     i. Введите логин, используемый для аутентификации.
     ii. Нажмите на кнопку **Выбрать файл ключа** и загрузите файл с ключом. Поддерживаются форматы RSA, DSA, ECDSA, ED25519.
     iii. Если публичный ключ защищен паролем, введите пароль в поле **Пароль закрытого ключа**.

6. Если необходимо повысить привилегии создаваемой учетной записи для администрирования сетевого оборудования, в разделе **Повышение привилегий** переведите переключатель, соответствующий используемой системе, в активное положение:

Сканируемая система	Уровень привилегий
Unix	Sudo
Cisco	Enable
CheckPoint	Expert
Huawei	Super

7. Проверьте корректность созданной учетной записи в разделе **Проверка учетной записи**:
   a. Выберите сканер в выпадающем списке.
   b. Укажите адрес устройства, к которому будет подключаться учетная запись, и порт подключения.

      Порт подключения по умолчанию — 22. Его можно изменить.

   c. Нажмите на кнопку **Проверить подключение**. Система проверит подключение с указанными параметрами и отобразит результат: **Аутентификация прошла успешно** или **Ошибка проверки учетной записи**.
8. Если учетная запись создана корректно и подключение к оборудованию проходит без ошибок, нажмите на кнопку **Добавить**. Учетная запись отобразится в списке записей.

Создание учетной записи типа Windows

Чтобы создать учетную запись типа **Windows**:

1. Нажмите на кнопку **Добавить** (+). Система отобразит выпадающий список возможных типов учетной записи.
2. Выберите тип **Windows**. Система отобразит окно добавления учетной записи Windows.
3. Выберите в выпадающем списке тенант, в котором будет создана учетная запись.
4. Укажите название и описание учетной записи.
5. В разделе **Параметры аутентификации** введите логин и пароль, используемые для аутентификации.

   Логин доменной учетной записи указывается в формате **login@domain**.

6. Проверьте корректность созданной учетной записи в разделе **Проверка учетной записи**:
   a. Выберите сканер в выпадающем списке.
   b. Укажите адрес устройства, к которому будет подключаться учетная запись.
   c. Нажмите на кнопку **Проверить подключение**. Система проверит подключение с указанными параметрами и отобразит результат: **Аутентификация прошла успешно** или **Ошибка проверки учетной записи**.
7. Если учетная запись создана корректно и подключение к оборудованию проходит без ошибок,

нажмите на кнопку **Добавить**. Учетная запись отобразится в списке записей.

Создание учетной записи типа СУБД

Чтобы создать учетную запись типа **СУБД**:

1. Нажмите на кнопку **Добавить** (+). Система отобразит выпадающий список возможных типов учетной записи.
2. Выберите тип учетной записи **СУБД**. Система отобразит окно добавления учетной записи СУБД.
3. Выберите в выпадающем списке тенант, в котором будет создана учетная запись.
4. Выберите в выпадающем списке тип системы управления базами данных, к которой необходимо подключиться.

   Доступные типы СУБД:

   - MS SQL;
   - Oracle;
   - PostgreSQL;
   - ClickHouse.
5. Введите название и описание учетной записи СУБД.

   Набор параметров аутентификации, отображаемых в окне добавления учетной записи, зависит от выбранного типа СУБД.

   - Для типа **MS SQL**
     a. Укажите экземпляр базы данных. Если он не указан, будет использоваться экземпляр по умолчанию.
     b. Введите логин и пароль, используемые для аутентификации.
     c. Введите название базы данных. Если она не указана, будет использоваться БД по умолчанию.
     d. Укажите номер порта, по которому производится подключение. По умолчанию используется порт 1433.
   - Для типа **Oracle**
     a. Укажите имя службы. Если оно не указано, будет использоваться имя службы по умолчанию.
     b. Введите логин и пароль, используемые для аутентификации.
     c. Введите название базы данных. Если она не указана, будет использоваться БД по умолчанию.
     d. Укажите номер порта, по которому производится подключение. По умолчанию используется порт 1521.
   - Для типа **PostgreSQL**
     a. Введите логин и пароль, используемые для аутентификации.
     b. Введите название базы данных. Если она не указана, будет использоваться БД по

умолчанию.

c. Укажите номер порта, по которому производится подключение. По умолчанию используется порт 5432.

- Для типа **ClickHouse**

a. Введите логин и пароль, используемые для аутентификации.

b. Введите название базы данных. Если она не указана, будет использоваться БД по умолчанию.

c. Укажите номер порта, по которому производится подключение. По умолчанию используется порт 8123.

6. Нажмите на кнопку **Проверить подключение**. Система проверит подключение с указанными параметрами и отобразит результат: **Аутентификация прошла успешно** или **Ошибка проверки учетной записи**.
7. Если учетная запись создана корректно и подключение к базе данных проходит без ошибок, нажмите на кнопку **Добавить**. Учетная запись отобразится в списке записей.

Просмотр учетной записи

Чтобы просмотреть учетную запись, нажмите на строку учетной записи в списке. В правой части экрана отобразится карточка этой учетной записи, содержащая подробную информацию о ней. Название учетной записи отображается в верхней части карточки.

Чтобы закрыть карточку, нажмите на крестик в правом верхнем углу карточки.

Изменение учетной записи

Чтобы изменить учетную запись:

1. Нажмите на строку учетной записи в списке. В правой части экрана отобразится карточка этой учетной записи.
2. Нажмите на кнопку **Изменить** (✎) на панели инструментов карточки учетной записи. Отобразится окно изменения учетной записи.
3. Внесите изменения в требуемые поля.
4. Нажмите на кнопку **Сохранить**. Измененные данные учетной записи будут сохранены.

Удаление учетной записи

Чтобы удалить учетную запись:

1. Нажмите на строку учетной записи в списке. В правой части экрана отобразится карточка этой учетной записи.
2. Нажмите на кнопку **Удалить** (🗑) на панели инструментов карточки учетной записи. Отобразится окно подтверждения удаления учетной записи.
3. Нажмите на кнопку **Удалить**. Система отобразит уведомление об удалении выбранной учетной записи, и запись исчезнет из списка учетных записей.

Проверка учетной записи

Чтобы проверить, можно ли с помощью учетной записи авторизоваться на хосте:

1. Нажмите на строку учетной записи в списке. В правой части экрана отобразится карточка этой учетной записи.
2. В разделе **Проверка учетной записи** выберите сканер из выпадающего списка.
3. Введите адрес и порт устройства, на котором необходимо проверить авторизацию, используя текущую учетную запись.

   Используемые по умолчанию номера портов для подключения:

   - Для учетной записи типа **SSH** — 22 (редактируется).
   - Для учетной записи типа **Windows** — 5985 (не редактируется).
   - Для учетной записи типа **СУБД** (редактируются):
     - PostgreSQL — 5432;
     - MSQL — 1433;
     - Oracle — 1521;
     - Clickhouse — 8123.
4. Нажмите на кнопку **Проверить подключение**. Система проверяет подключение с указанными параметрами и отображает результат: **Аутентификация прошла успешно** или **Ошибка проверки учетной записи**.

Профили сканирования

Управление профилями сканирования осуществляется в разделе **Сканирование** → **Профили**.

О профилях сканирования

Профили сканирования содержат данные о технических параметрах сканирования узлов и сетей. В системе доступны несколько предустановленных типов профилей. На их основе можно создавать пользовательские профили, настраивать параметры профилей, сгруппированные по вкладкам, и выполнять следующие задачи:

Тип профиля	Выполняемые задачи	Доступные вкладки
Базовое сканирование	Сканирование сети с целью обнаружения активных устройств и сбора базовой инвентарной информации: например, установленное ПО, пользователи и открытые порты	• Основное • Настройки обнаружения • Аутентификация • Импорт дополнительных данных

Тип профиля	Выполняемые задачи	Доступные вкладки
Поиск уязвимостей	Сканирование сети с целью обнаружения активных устройств, сбора базовой инвентарной информации и поиска уязвимостей	• Основное • Настройки обнаружения • Аутентификация • Импорт дополнительных данных
Обнаружение	Сканирование сети с целью обнаружения активных устройств и открытых портов	• Основное • Настройки обнаружения • Импорт дополнительных данных
Тестирование на проникновение	Сканирование сети с целью обнаружения активных сервисов и поиска уязвимостей с помощью безопасных и небезопасных проверок	• Основное • Настройки обнаружения • Тестирование на проникновение • Безопасные проверки • Небезопасные проверки • Подбор паролей
Проверка стандартов	Сканирование сети с целью проверки соответствия системы установленным нормативным требованиям и стандартам безопасности	• Основное • Настройки обнаружения • Аутентификация • Настройки стандартов • Импорт дополнительных данных

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (＋) позволяет создать новый профиль сканирования.
- Поле **Поиск** предназначено для быстрого поиска профилей в отображаемом списке по полю **Название**.

- Кнопка **Фильтр** ( ) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка объектов.

В рабочей области отображается таблица имеющихся профилей, которая содержит следующие столбцы:

- **Название** профиля.
- **Описание** профиля.
- **Тип** профиля.
- **Режим сканирования**.
- **Тенант**.
- **Дата обновления**.
- **Дата создания**.

При работе с таблицей профилей доступны следующие операции:

- Поиск профилей по полю **Название**.
- Фильтрация задач по полям **Тенант**, **Тип** и **Режим**.
- Сортировка задач по полям **Название**, **Описание**, **Дата создания** и **Дата обновления**.

При выборе конкретного профиля на экране открывается страница с детальной информацией о нем.

Работа с профилем сканирования

Доступные операции над профилем:

- Создание профиля сканирования
- Просмотр профиля сканирования
- Изменение профиля сканирования
- Удаление профиля

Создание профиля сканирования

Чтобы создать профиль:

1. Нажмите на кнопку **Создать** ( ). Отобразится форма создания профиля.
2. Выберите режим сканирования: **Безагентское** или **Агентское**.
3. Выберите тип, используемый для создания профиля.
4. Выберите из выпадающего списка тенант, в котором будет создан профиль.
5. Введите название и описание профиля.
6. Нажмите на кнопку **Создать**. Созданный профиль выбранного типа отобразится в списке профилей и станет доступен для настройки.

Просмотр профиля сканирования

Чтобы просмотреть профиль, нажмите на строку профиля в списке. Система отобразит страницу с подробной информацией о нем.

Название профиля отображается в верхней части экрана.

Панель инструментов страницы включает в себя следующие компоненты:

- **Изменить** ( ) — позволяет изменить профиль.
- **Удалить** ( ) — позволяет удалить профиль.

Страница содержит вкладки профиля. Отображаемый набор вкладок зависит от типа, на основе которого создан профиль сканирования.

Изменение профиля сканирования

Чтобы изменить профиль сканирования:

1. Нажмите на строку профиля в списке. Система отобразит страницу этого профиля. В левой части отображается список вкладок с параметрами профиля.
2. Нажмите на кнопку **Изменить** ( ). Вкладки профиля станут доступными для редактирования.
3. Отредактируйте вкладки профиля.
4. Нажмите на кнопку **Сохранить**. Изменения будут сохранены.

Удаление профиля

1. Нажмите на строку профиля в списке. Система отобразит страницу этого профиля. В левой части отображается список вкладок с параметрами профиля.
2. Нажмите на кнопку **Удалить** ( ) на панели инструментов страницы профиля. Отобразится окно подтверждения удаления профиля.
3. Нажмите на кнопку **Удалить**. Система отобразит уведомление об удалении выбранного профиля, и профиль исчезнет из списка профилей.

Вкладки профилей

Настройка профилей производится с помощью следующих вкладок:

- **Основное** — основная информация о сканировании.
- **Настройки обнаружения** — настройки сканирования сети с целью обнаружения активных устройств и открытых портов.
- **Аутентификация** — настройки портов, к которым подключается учетная запись.
- **Импорт дополнительных данных** — настройки импорта сведений об открытых портах.
- **Тестирование на проникновение** — настройки сканирования сети с помощью безопасных и небезопасных проверок.
- **Безопасные проверки** — настройки скриптов, проверяющих сведения информативного характера.

- **Небезопасные проверки** — настройки скриптов, которые могут причинить вред оборудованию при эксплуатации.
- **Подбор паролей** — настройки подбора паролей для получения несанкционированного доступа к системам и учетным записям.
- **Настройки стандартов** — настройки стандартов безопасности, применяемых в задаче сканирования.

Список доступных вкладок определяется типом профиля.

Вкладка Основное

Вкладка **Основное** включает в себя следующие поля:

- **Тенант**.
- **Название** профиля.
- **Режим сканирования**.
- **Тип** профиля.

Вкладка Настройки обнаружения

Вкладка **Настройки обнаружения** содержит настройки обнаружения, выбранные для профиля. Изменить настройки можно при редактировании профиля с помощью кнопки **Изменить** (✎).

Существуют следующие режимы работы настроек:

- **Стандартный** — настройки выполняются с использованием ограниченного числа полей и переключателей, доступных в интерфейсе системы.
- **Пользовательский** — настройки выполняются с использованием всех доступных параметров командной строки.

Стандартные настройки

Обнаружение хостов

Чтобы выполнить обнаружение хостов:

1. Укажите порты для TCP SYN пингования — отправки пустого TCP пакета с установленным SYN флагом на заданные порты.

   Отдельные порты и диапазоны указываются в этом и следующем поле через запятую.

   Пример: 22,80,100-1000.

2. Укажите порты для UDP пингования — отправки пустого UDP пакета на заданные порты.
3. Установите следующие переключатели в активное положение, чтобы разрешить:
   - ICMP пингование;
   - DNS-имена устройств обнаруженных IP-адресов;

- Автоматическое определение ОС.

Включение этого и следующего параметров увеличивает время сканирования.

- Автоматическое определение версии сервисов.

4. Укажите шаблон настроек времени:
   - T0 — очень медленный;
   - T1 — медленный;
   - T2 — умеренный;
   - T3 — стандартный;
   - T4 — быстрый;
   - T5 — очень быстрый.
5. Укажите таймаут соединения в минутах.

Сканирование портов

Чтобы выполнить сканирование портов:

1. Укажите порты для TCP SYN пингования.

   Отдельные порты и диапазоны указываются в этом и следующем поле через запятую.

   Пример: 22,80,100-1000

2. Укажите порты для UDP пингования.

Расширенные настройки

Система отображает поле командной строки, где вы можете указать все требуемые параметры.

Вкладка Аутентификация

Вкладка **Аутентификация** включает в себя следующие поля:

- **Порты SSH** — порты, к которым будет подключаться учетная запись типа **SSH**.
- **Порты Windows** — порт, к которому будет подключаться учетная запись типа **Windows**. Поле недоступно для редактирования.

Вкладка Импорт дополнительных данных

Вкладка **Импорт дополнительных данных** содержит переключатель **Открытые порты**. Переведите его в активное положение, чтобы разрешить импорт сведений об открытых портах.

Вкладка Тестирование на проникновение

Вкладка **Тестирование на проникновение** становится доступной при создании и редактировании профиля типа **Тестирование на проникновение**. На этой вкладке отображается выбранный для

профиля тип сканирования портов. Изменить тип сканирования можно при редактировании профиля с помощью кнопки **Изменить** ( ).

Доступные типы сканирования портов приведены в таблице:

Тип сканирования	Описание
Базовый	TCP: 21, 22, 23, 25, 53, 80, 110, 139, 143,161, 389, 443, 445, 465, 515, 554, 631, 587, 993, 995, 1433, 1521, 1702, 1723, 3128, 3268, 3306, 3389, 5432, 5060, 5061, 5900, 5985, 5986, 8080; UDP: 20, 53, 67, 68, 69, 88, 123,135,137,138, 161, 162, 389.
Топ 25	25 самых распространенных портов TCP и UDP
Топ 100	100 самых распространенных портов TCP и UDP
Все порты	Все существующие порты TCP и UDP от 0 до 65535 Сканирование всех портов может занять длительное время и влиять на производительность сети. Этот тип сканирования не рекомендуется для подсетей с длиной маски 22 и более бит.
Пользовательский список	Список портов TCP и UDP задается пользователем. Можно задавать как определенные порты через запятую, так и диапазоны. Пример: 22,80,100-1000.

Вкладка Безопасные проверки

Вкладка **Безопасные проверки** становится доступной при создании и редактировании профиля типа **Тестирование на проникновение**.

К скриптам для безопасных проверок относятся скрипты, проверяющие сведения информативного характера: например, сведения о сертификатах, небезопасных сервисах и так далее.

На вкладке отображается информация об использовании безопасных проверок при тестировании, а также название режима использования. Если переключатель на вкладке активирован, при сканировании будут использоваться только указанные скрипты безопасных проверок.

Изменить настройки вкладки можно при редактировании профиля с помощью кнопки **Изменить** ( ).

Настройте параметры вкладки:

1. Переведите переключатель **Использовать безопасные проверки** в активное положение.
2. Выберите режим проверок:
   - **Все проверки** — система будет использовать для проверок все скрипты.
   - **Пользовательский** — система будет использовать для проверок только скрипты, выбранные пользователем.

Если выбран режим **Пользовательский**, система отобразит таблицу выбора скриптов, где с помощью кнопок ← и → нужно сформировать в столбце **Используемые** список скриптов для проверок. Для выбора скрипта в списке установите флажок рядом с его названием. В поле поиска над таблицей можно выполнить поиск по названию скрипта.

Чтобы перенести сразу все скрипты в нужный столбец, используйте кнопки ↞ и ↠.

Вкладка Небезопасные проверки

Вкладка **Небезопасные проверки** становится доступной при создании и редактировании профиля типа **Тестирование на проникновение**.

К скриптам для небезопасных проверок относятся скрипты, которые могут причинить вред оборудованию при эксплуатации: например, использование эксплойтов или DDoS.

На вкладке отображается информация об использовании небезопасных проверок при тестировании, а также название режима использования. Если переключатель на вкладке активирован, при сканировании будут использоваться только указанные скрипты небезопасных проверок.

Изменить настройки вкладки можно при редактировании профиля с помощью кнопки **Изменить** ( ).

Настройте параметры вкладки:

1. Переведите переключатель **Использовать небезопасные проверки** в активное положение.
2. Выберите режим проверок:
   - **Все проверки** — система будет использовать для проверок все скрипты.
   - **Пользовательский** — система будет использовать для проверок только скрипты, выбранные пользователем.

     Если выбран режим **Пользовательский**, система отобразит таблицу выбора скриптов, где с помощью кнопок ← и → нужно сформировать в столбце **Используемые** список скриптов для проверок. Для выбора скрипта в списке установите флажок рядом с его названием. В поле поиска над таблицей можно выполнить поиск по названию скрипта.

     Чтобы перенести сразу все скрипты в нужный столбец, используйте кнопки ↞ и ↠.

Вкладка Подбор паролей

Вкладка **Подбор паролей** становится доступной при создании и редактировании профиля типа **Тестирование на проникновение**. На этой вкладке отображаются сервисы, для которых запускается функционал подбора паролей, словари учетных записей и паролей, сведения о порте подключения, параметры, отображаемые в зависимости от выбранного сервиса, а также дополнительные настройки этого функционала.

Изменить настройки на вкладке можно при редактировании профиля с помощью кнопки **Изменить** ( ).

По умолчанию система использует для подбора паролей собственные словари учетных записей и паролей. При настройке конфигурации можно загрузить в систему пользовательские словари.

Поля, доступные для редактирования на этой вкладке, зависят от выбранного сервиса.

Чтобы настроить конфигурацию подбора паролей, выполните следующие действия:

1. Для загрузки в систему пользовательского словаря учетных записей в разделе **Словарь учетных записей** нажмите на кнопку **Загрузить файл словаря**. Данные должны быть внесены в файл формата TXT с использованием переноса строки в качестве разделителя.
2. Для загрузки в систему пользовательского словаря паролей в разделе **Словарь паролей** нажмите на кнопку **Загрузить файл словаря**. Данные должны быть внесены в файл формата TXT с использованием переноса строки в качестве разделителя.
3. В поле **Порт назначения** отображается номер порта, используемого по умолчанию в качестве стандартного порта сервиса, выбранного в настоящий момент. Чтобы использовать другой порт, переведите переключатель **Нестандартный порт** в активное положение и укажите номер порта в поле **Порт назначения**.
4. Переведите нужные переключатели в активное положение для включения дополнительных параметров:
   - **Проверять пустые пароли**.
   - **Проверять логин в качестве пароля** — позволяет проверить комбинации, когда логин и пароль совпадают.
   - **Завершать проверку узла после совпадения** — завершает проверку узла после обнаружения первого совпадения пары "логин + пароль".
5. Укажите в поле **Параллельный перебор** количество одновременно обрабатываемых логинов.

Вкладка Настройки стандартов

Вкладка **Настройки стандартов** становится доступной при создании и редактировании профиля типа **Проверка стандартов**. На этой вкладке отображается список стандартов безопасности, применяемых в задаче сканирования. Вы можете выполнить проверку системы на соответствие стандартам безопасности, выбранным из списка доступных стандартов.

Файл, содержащий данные для проверки соответствия, можно обновить с помощью менеджера обновлений.

Чтобы выбрать стандарты безопасности, которые должны применяться в задаче сканирования:

1. Просмотрите отображающийся в таблице список доступных в системе стандартов безопасности. При необходимости воспользуйтесь полем поиска или функцией пагинации.
2. Переведите переключатели рядом с названиями стандартов в активное положение, чтобы выбрать эти стандарты для сканирования.

По нажатии кнопки **Выбрать все** можно выбрать сразу все стандарты безопасности, доступные в системе.

Задачи сканирования

Задачи сканирования создаются на основе выбранного режима сканирования и различных настроек.

Работа с задачами сканирования осуществляется в разделе **Сканирование** → **Задачи**.

- Безагентское сканирование
- Агентское сканирование
- Настройка отображения событий аудита, связанных со сканированием

Безагентское сканирование

Задачи сканирования создаются на основе выбранных настроек профиля и указанной учетной записи. Они позволяют сканировать оборудование и сети для инвентаризации оборудования и поиска уязвимостей на оборудовании с указанной периодичностью.

Работа с задачами сканирования осуществляется в разделе **Сканирование** → **Задачи**.

Интерфейс раздела

Панель инструментов включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новую задачу.
- Поле **Поиск** предназначено для быстрого поиска задач в отображаемом списке по имени.
- Кнопка **Фильтр** ( ) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка задач.

В рабочей области раздела отображается таблица имеющихся задач сканирования. Таблица содержит следующие столбцы:

- **Статус** — статус задачи. Доступные статусы:
  - **Не выполнялось**
  - **Выполняется**
  - **Приостановлено**
  - **Ошибка**
  - **Отменено**
  - **Выполнено**
- **Наименование** — наименование задачи.
- **Режим сканирования** — режим сканирования, в котором работает задача.
- **Профиль** — профиль, на основе которого создана задача.
- **Тенант** — тенант, в котором создана задача.
- **Дата обновления** — дата последнего обновления задачи.
- **Дата создания** — дата создания задачи.
- **Следующий запуск** — информация о способе выполнения следующего запуска задачи. Если следующий запуск не может быть выполнен, в столбце отображается значение **Запуск невозможен**.

При работе с таблицей задач доступны следующие операции:

- Поиск задач по полю **Наименование**.
- Фильтрация задач по полям **Тенант**, **Статус**, **Тип профиля** и **Режим сканирования**.
- Сортировка задач по полям **Наименование**, **Дата создания** и **Дата обновления**.
- Настройка отображения таблицы.

При выборе конкретной задачи на экране открывается страница с детальной информацией о ней.

Работа с задачей безагентского сканирования

Доступные операции над задачей:

- Создание задачи безагентского сканирования
- Просмотр задачи безагентского сканирования

Создание задачи безагентского сканирования

Чтобы создать задачу, выполните следующие действия:

1. Перейдите в раздел **Сканирование** → **Задачи**.
2. Нажмите на кнопку **Создать** (＋). Отобразится окно создания задачи.
3. Выберите из выпадающего списка тенант, в котором будет создана задача.
4. Укажите название и описание задачи.
5. Выберите из выпадающего списка **Режим сканирования** значение **Безагентское**.
6. Выберите из выпадающих списков следующие значения:
   a. Профиль сканирования, параметры которого будут использоваться в задаче;
   b. Сканер, собирающий сведения об оборудовании.
7. Выберите учетные записи, которые будут использоваться для аутентификации на оборудовании при сканировании:
   a. Нажмите на кнопку **Добавить учетные записи** (＋). Система отобразит окно добавления учетных записей со списком доступных записей.
   b. Установите флажок для выбора хотя бы одной учетной записи.
   c. Нажмите на кнопку **Добавить**.

Если выбран профиль сканирования типа **Тестирование на проникновение**, выбор учетных записей недоступен.

8. В разделе **Цели** перечислите сетевые адреса или диапазоны адресов, которые требуется просканировать.
9. В разделе **Исключения** перечислите сетевые адреса или диапазоны адресов, исключаемые из сканирования.

Поля для перечисления адресов можно расширить, потянув их за правый нижний угол. Это позволит увидеть все введенные данные одновременно.

10. Задайте расписание запуска задачи, выбрав из списка нужную периодичность запуска.

    Если расписание запуска задачи не настроено, все действия в задаче (запуск, приостановку, возобновление и остановку запуска) можно выполнять вручную.

    Если расписание запуска задачи настроено, действия запуска, приостановки и возобновления запуска выполняются только системой согласно расписанию. Вручную можно только остановить задачу.

11. Для всех значений периодичности, кроме **Вручную** и **Однократно**, укажите временной интервал запуска в полях **Срок действия**:
    a. дата и время первого запуска задачи;
    b. дата и время, после которых задача больше не будет запускаться.
12. Для всех значений периодичности, кроме **Вручную** и **Однократно**, укажите режим повторов запуска:
    - Для режима **Ежедневно** — интервал повтора в днях.
    - Для режима **Еженедельно** — интервал повтора в неделях с указанием дней недели.
    - Для режима **Ежемесячно** — интервал повтора в месяцах с указанием дней месяца.
13. При необходимости для всех значений периодичности, кроме **Вручную**, можно указать интервал времени, в течение которого все запуски задачи будут приостановлены. Для этого выполните следующие действия:
    a. Установите флажок **Приостановка работы**. Система отобразит настройки приостановки запусков.
    b. Выберите дни недели приостановки запусков.
    c. Укажите время начала приостановки.
    d. Укажите длительность приостановки.
14. В разделе **Дополнительные параметры** укажите режим журналирования задачи:
    - **Информационный** — в этом режиме система сохраняет в хранилище событий список ошибок, возникших при запуске задачи.
    - **Расширенный** — помимо списка ошибок, в этом режиме система сохраняет в хранилище событий полную информацию о процессе сканирования: логи сервисов, данные о подключении к оборудованию, попытках авторизации, попытках выполнения команд, результатах выполнения команд и отключении от хоста.
15. Завершите процесс создания задачи:
    - Если необходимо создать задачу, не запуская ее, нажмите на кнопку **Создать**.
    - Если необходимо запустить задачу сразу после ее создания, нажмите на кнопку **Создать и запустить**.

Чтобы отменить процесс создания задачи, нажмите на кнопку **Отменить**.

Просмотр задачи безагентского сканирования

Для просмотра страницы задачи нажмите на запись задачи в списке. На странице задачи доступны

следующие действия:

Кнопка на странице задачи	Действие	Комментарии
Запустить	Запустить выполнение задачи	Начинается сканирование указанных целей. Статус задачи изменяется на **Выполняется**.
Изменить ( )	Изменить параметры задачи в режиме редактирования	
	Удалить задачу из системы	
	Приостановить выполнение задачи	Сканирование приостанавливается. Статус задачи изменяется на **Приостановлено**.
Возобновить сканирование	Возобновить выполнение задачи	Сканирование возобновляется. Статус задачи изменяется на **Выполняется**.
	Остановить выполнение задачи	Сканирование полностью останавливается. Статус задачи изменяется на **Отменено**.

Также на странице задачи доступны вкладки **Параметры** и **История запусков**.

Параметры

Для просмотра сведений о параметрах задачи сканирования перейдите на вкладку **Параметры** страницы задачи.

История запусков

Если выбрана вкладка **История запусков**, в левой части экрана отображается список всех запусков задачи. Система создает отдельный экземпляр задачи в списке **История запусков** при каждом ручном запуске из интерфейса системы или запуске по расписанию.

Каждый запуск при выполнении проходит этапы согласно профилю сканирования, указанному в задаче. Этапы состоят из подэтапов:

Этап	Подэтап
Обнаружение	Обнаружение
Сканирование оборудования	Инвентаризация
	Сбор данных
	Поиск уязвимостей
Тестирование на проникновение	Обнаружение
	Тестирование на проникновение

Таблица списка запусков содержит следующие столбцы:

- **Статус** — статус запуска.
- **Запуск** — дата и время запуска.

При работе с таблицей запусков доступны следующие операции:

- Фильтрация этапов по полю **Статус**.
- Сортировка этапов по полю **Статус**.

Статусы запусков:

- **Выполняется** — запущен пользователем или по расписанию и выполняется в настоящий момент.
- **Отменен** — остановлен пользователем.
- **Завершен** — все подэтапы этого запуска имеют статус **Завершен**.
- **Ошибка** — хотя бы в одном этапе запуска имеется статус **Ошибка**.
- **Приостановлен** — выполнение запуска задачи приостановлено пользователем или по расписанию.

В центральной части экрана для каждого запуска можно просмотреть статистику по целям, найденным устройствам, результатам и ошибкам сканирования, а также длительности сканирования. Эта длительность рассчитывается как интервал от времени первого старта запуска до итогового времени окончания запуска.

Ниже отображается таблица этапов сканирования оборудования. Она содержит следующие столбцы:

- **Статус** — статус запуска.
- **Этап** — дата и время запуска.
- **Цель** — просканированные сетевые адреса.

При работе с таблицей этапов доступны следующие операции:

- Поиск этапов по полю **Цель**.
- Фильтрация этапов по полю **Статус**.
- Сортировка этапов по полям **Статус** и **Цель**.

Статусы этапов сканирования:

- **Ожидает** — этап ожидает выполнения.
- **В работе** — этап выполняется прямо сейчас.
- **Отменен** — этап прерван из-за прерывания запуска пользователем.
- **Завершен** — все подэтапы этапа выполнены успешно.
- **Ошибка** — хотя бы у одного подэтапа этапа имеется статус **Ошибка**.

Просмотр событий сканирования этапа

Вы можете просмотреть события этапа сканирования в хранилище событий, чтобы оценить,

насколько успешно просканировалось оборудование.

Хранимые события зависят от режима журналирования, установленного в дополнительных параметрах задачи (**Информационный** или **Расширенный**). Хранилище может быть пустым, если в профиле сканирования был указан порт или список портов, а на оборудовании не был открыт ни один из этих портов, то есть оборудование было проигнорировано при сканировании.

Для корректной работы функции перехода к списку событий требуется предварительная настройка. Данная настройка выполняется только при первоначальном запуске или обновлении системы.

Чтобы перейти на страницу со списком событий сканирования:

1. Наведите курсор на строку этапа. В правой части строки отобразится кнопка .
2. Нажмите на кнопку . Система откроет раздел с автоматически выбранным хранилищем событий этапа сканирования.

Агентское сканирование

Задачи агентского сканирования создаются на основе выбранных настроек профиля. Они позволяют сканировать оборудование, на котором установлены агенты, посредством модуля агентов платформы и получать информацию об этом оборудовании.

При подготовке агентского сканирования оборудования под управлением Windows можно использовать только следующие политики выполнения PowerShell:

- **Bypass**;
- **RemoteSigned**;
- **Unrestricted**.

При использовании других политик агент не сможет запускать команды.

Работа с задачами сканирования осуществляется в разделе **Сканирование** → **Задачи**.

Интерфейс раздела

Панель инструментов включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новую задачу.
- Поле **Поиск** предназначено для быстрого поиска задач в отображаемом списке по имени.
- Кнопка **Фильтр** ( ) позволяет открыть в правой части рабочей области панель для настройки фильтрации списка задач.

В рабочей области раздела отображается таблица имеющихся задач. Таблица содержит следующие столбцы:

- **Статус** — статус задачи. Доступные статусы:
  - **Не выполнялось**;
  - **Выполняется**;

- **Ошибка**;
- **Выполнено**;
- **Отменено**;
- **Приостановлено**.

- **Наименование** — наименование задачи.
- **Режим сканирования** — режим сканирования, в котором работает задача.
- **Профиль** — профиль, на основе которого создана задача.
- **Тенант** — тенант, в котором создана задача.
- **Дата обновления** — дата последнего обновления задачи.
- **Дата создания** — дата создания задачи.
- **Следующий запуск** — информация о времени и способе выполнения следующего запуска задачи. Если следующий запуск не может быть выполнен, в столбце отображается значение **Запуск невозможен**.

При работе с таблицей задач доступны следующие операции:

- Поиск задач по полю **Наименование**.
- Фильтрация задач по полям **Тенант**, **Статус**, **Тип профиля** и **Режим сканирования**.
- Сортировка задач по полям **Наименование**, **Дата создания** и **Дата обновления**.

При выборе конкретной задачи на экране открывается страница с детальной информацией о ней.

Работа с задачей агентского сканирования

Доступные операции над задачей:

- Создание задачи агентского сканирования
- Просмотр задачи агентского сканирования

Создание задачи агентского сканирования

Чтобы создать задачу, выполните следующие действия:

1. Перейдите в раздел **Сканирование** → **Задачи**.
2. Нажмите на кнопку **Создать** (➕). Отобразится окно создания задачи.
3. Выберите из выпадающего списка тенант, в котором будет создана задача.
4. Укажите название и описание задачи.
5. Выберите режим сканирования: **Агентское**.
6. Выберите профиль сканирования, параметры которого будут использоваться в задаче.
7. Добавьте группы агентов в разделе **Цели**. Для этого:
   a. Нажмите на кнопку **Добавить группы агентов**. Система отобразит окно добавления групп агентов, доступных для выполнения сканирования.
   b. Укажите хотя бы одну группу агентов, связанную с задачей.

c. Нажмите на кнопку **Добавить**.

8. Нажмите на кнопку **Создать**. Задача сканирования будет создана.

После создания задачи система будет запускать сбор данных с агентов автоматически каждые 3 часа. Конфигурация расписания запуска задачи вручную недоступна.

Чтобы отменить процесс создания задачи, нажмите на кнопку **Отменить**.

Просмотр задачи агентского сканирования

Для просмотра задачи нажмите на запись задачи в списке. Доступны следующие действия:

Кнопка на странице задачи	Действие
Изменить ( )	Изменить параметры задачи в режиме редактирования
	Удалить задачу из системы

Также на странице задачи доступны вкладки **Параметры** и **Статус**.

Параметры

Вкладка **Параметры** страницы задачи позволяет просмотреть сведения о параметрах задачи сканирования.

Статус

На вкладке **Статус** система отображает таблицу со списком оборудования, просканированного в рамках этой задачи. Таблица содержит следующие столбцы:

- **Статус** — сведения о статусе оборудования на основе результатов последнего сканирования. Доступные статусы:
  - **Не выполнялось** — оборудование создано в задаче, но сканирование еще не выполнялось.
  - **Выполняется** — выполняется получение и обработка результатов сканирования.
  - **Выполнено** — сканирование успешно завершено.
  - **Ошибка** — в процессе обработки результатов возникла ошибка.
  - **Отменено** — сканирование отменено. Возможные причины отмены:
    - оборудование больше не принадлежит ни к одной группе агентов;
    - оборудование было исключено из группы агентов, указанной в задаче сканирования;
    - группа агентов была удалена из задачи сканирования.
- **Название** — название оборудования.
- **Группа** — группа агентов, к которой относится агент, установленный на оборудовании.
- **IP-адрес** — сведения об IP-адресах оборудования.
- **ОС** — операционная система, под которой работает оборудование.

- **Дата последнего сканирования** — дата последнего сканирования оборудования системой.

При работе с таблицей оборудования доступна сортировка объектов по полям **Статус**, **Название**, **ОС** и **Дата последнего сканирования**.

Настройка отображения событий аудита, связанных со сканированием

Данный раздел описывает процесс настройки отображения событий аудита, отфильтрованных согласно критериям этапа сканирования устройства.

Настройка выполняется только при первоначальном запуске или обновлении системы.

О разделе настройки

Сервис аудита регистрирует и сохраняет события каждого этапа сканирования оборудования в системном хранилище аудита. Вы можете просмотреть все события, связанные с этапом, перейдя из задачи сканирования в раздел **Поиск**. Система автоматически отобразит события в этом разделе.

Для корректного отображения событий этапа сканирования необходимо выполнить настройку перехода к списку событий в хранилище.

Интерфейс раздела настройки

В интерфейсе раздела доступно поле **Шаблон фильтров**, которое позволяет создать шаблон, позволяющий корректно отобразить в разделе **Поиск** отфильтрованные события аудита для этапа сканирования.

- Кнопка позволяет сделать поле **Шаблон фильтров** доступным для редактирования.
- Кнопка позволяет отправить данные адресной строки раздела **Поиск** на сервер системы для создания шаблона фильтров.
- Кнопка позволяет отменить вставку адресной строки в поле **Шаблон фильтров** и выйти из режима редактирования поля.

Работа с разделом настройки

Чтобы выполнить настройку:

1. Перейдите в раздел **Поиск**.
2. Выберите из выпадающего списка в верхней части раздела хранилище логов сканирования.
3. Нажмите на кнопку **Добавить фильтр** ( ). Отобразится окно добавления фильтра.
4. Выберите из выпадающего списка поле `hostid`.
5. Введите в поле **Значение** любое значение.
6. Нажмите на кнопку **Сохранить**.
7. Повторите шаги 2—5, чтобы добавить поле `scanid`.
8. Нажмите на кнопку на панели инструментов и выберите из выпадающего списка способ задания периода **Задать период**.

9. Укажите любые даты начала и конца периода.
10. Скопируйте адресную строку браузера.
11. Перейдите в раздел **Настройки** → **Сканеры**.
12. Нажмите на кнопку . Поле для вставки ссылки станет доступным для редактирования.
13. Вставьте скопированную адресную строку в поле.
14. Нажмите на кнопку . Используя компоненты вставленной ссылки, система создаст шаблон фильтров, который позволит переходить из строки этапа сканирования в задаче к списку событий аудита, отфильтрованных по данным этапа.

    Если действие вставки ссылки необходимо отменить, нажмите на кнопку .

Настройка завершена. Теперь вы можете просматривать список событий выбранного этапа сканирования согласно описанию в разделе Просмотр событий сканирования этапа.

Проверки соответствия

В данном разделе описана работа с проверками соответствия, а также с базой данных, содержащей сведения о проверках.

Содержание раздела:

- Проверки
- База проверок

Проверки

Данный раздел описывает процесс работы с проверками соответствия в системе. Работа осуществляется в разделе **Проверки соответствия** → **Проверки** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О проверках соответствия

Проверки соответствия представляют собой массив записей с информацией, полученной при запуске задач сканирования на основе профиля типа **Проверка стандартов**. Эти задачи используются для сканирования сети с целью проверки соответствия системы установленным нормативным требованиям и стандартам безопасности.

Файл, содержащий данные для проверки соответствия, можно обновить с помощью менеджера обновлений.

Интерфейс раздела

Панель инструментов включает в себя следующие компоненты:

- Поле **Поиск** предназначено для быстрого поиска проверок стандартов в отображаемом списке

по идентификатору.

- Кнопка **Удалить** позволяет удалить запись проверки стандартов. Кнопка становится доступной после выбора хотя бы одной записи из списка.

В рабочей области отображается таблица имеющихся записей проверок стандартов. При выборе конкретной проверки стандартов на экране открывается карточка с детальной информацией о ней.

Таблица содержит следующие столбцы:

- **Статус** — статус выполненной проверки. Доступные статусы:
  - **Пройдена** — проверка успешно пройдена.
  - **Не пройдена** — проверка не пройдена.
  - **Неприменима** — проверку выполнить не удалось (например, из-за несоответствия ОС, для которой разработан стандарт проверки, и ОС на сканируемом устройстве).
- **Критичность** — уровень критичности проверки. Доступные уровни:
  - **Высокий**;
  - **Средний**;
  - **Низкий**.
- **ID** — уникальный идентификационный код проверки.
- **Название** — название проверки.
- **Стандарт** — название стандарта, в рамках которого проводилась проверка.
- **Операционная система** — сведения об операционной системе сканируемого устройства.
- **Дата проверки** — дата выполнения проверки.
- **Профиль** — название профиля, на основе которого выполнялась задача сканирования.
- **Имя устройства** — название сканируемого устройства.
- **Теги** — теги, позволяющие классифицировать проверку.
- **Категория** — категория проверки.

При работе с таблицей объектов доступны следующие операции:

- Поиск объектов по полю **ID**.
- Сортировка объектов по полям **Статус**, **ID** и **Дата проверки**.
- Настройка отображения таблицы.

При выборе конкретной проверки соответствия в правой части рабочей области отображается ее карточка с детальной информацией. Карточка проверки соответствия содержит следующие вкладки:

- **Основное** — содержит основные сведения о проверке.
- **Детали** — содержит сведения об описании проверки, решении и результате проверки.
- **Устройство** — содержит сведения о сканируемом устройстве.

Работа с проверками соответствия

Доступные операции над проверками соответствия:

- Просмотр проверки соответствия
- Изменение данных проверки соответствия
- Удаление проверки соответствия

Просмотр проверки соответствия

Чтобы просмотреть проверку соответствия:

1. Перейдите в раздел **Проверки соответствия** → **Проверки**. Система отобразит сведения об имеющихся проверках.
2. Нажмите на строку проверки в списке. Система отобразит страницу с карточкой этой проверки с подробной информацией о ней.

В верхней части карточки отображается ID проверки. Справа от ID расположена кнопка ⋮, по которой открывается выпадающее меню с действиями, доступными для выбранного объекта.

Чтобы закрыть карточку и вернуться к списку проверок, нажмите на крестик в правом верхнем углу карточки.

В карточке отображаются следующие вкладки:

- **Основное** — содержит основные сведения о проверке.
- **Детали** — содержит сведения об описании проверки, решении и результате проверки.
- **Устройство** — содержит сведения о сканируемом устройстве.

Вкладка Основное

На вкладке **Основное** отображаются основные сведения о проверке:

- **ID проверки** — уникальный идентификационный код проверки.
- **Статус** — статус проверки.
- **Название** — название проверки.
- **Категория** — категория проверки.
- **Стандарт** — список стандартов, к которым относится проверка. Список представлен в виде таблицы со столбцом **Значения**, в котором отображаются наименования стандартов. Наименования можно отсортировать. Вы можете развернуть таблицу с помощью кнопки ⌄ или свернуть с помощью кнопки ⌃.
- **Уровень критичности** — уровень критичности проверки.
- **Профиль** — название профиля, на основе которого выполнялась задача сканирования.
- **Дата проверки** — дата выполнения проверки.
- **Ссылки** — список ссылок на источники информации о проверке соответствия. Список представлен в виде таблицы со столбцом **Значения**, в котором отображаются ссылки. Ссылки можно отсортировать. Вы можете развернуть таблицу с помощью кнопки ⌄ или свернуть с

помощью кнопки ^ .

Вкладка Детали

Вкладка содержит следующие поля:

- **Проверка** — ID проверки.
- **Описание** — описание проверки.
- **Решение** — решение по проверке.
- **Результат проверки** — результат проверки.

Вкладка Устройство

Вкладка содержит следующие поля:

- **Название** — имя устройства, на котором проводилась проверка.
- **Критичность** — уровень критичности проверки.
- **Статус** — статус проверки.
- **Домен/Рабочая группа** — название домена или рабочей группы, в которых производилась проверка.
- **Операционная система** — сведения об операционной системе на сканируемом устройстве.
- **Тип узла** — тип сканируемого устройства.
- **Администратор безопасности** — имя администратора безопасности, контролирующего сканируемое устройство.
- **Сетевые интерфейсы** — список сетевых интерфейсов сканируемого устройства. Список представлен в виде таблицы со сведениями о сетевых интерфейсах. Сведения в таблице можно отсортировать по полю **IP-адрес**. Вы можете развернуть таблицу с помощью кнопки v или свернуть с помощью кнопки ^ .

  Таблица содержит следующие столбцы:

  - **IP-адрес** — IP-адрес интерфейса в сети.
  - **Маска сети** — диапазон IP-адресов, принадлежащих той же подсети, что и интерфейс.
  - **MAC-адрес** — физический (аппаратный) адрес сетевого интерфейса, уникальный для устройства.
- **Группы ИТ-активов** — столбец **Название** раздела содержит сведения о группах ИТ-активов, связанных с устройством.

Изменение данных проверки соответствия

В ходе изменения существующей проверки можно изменить сведения о проверке и сканируемом устройстве.

Изменение проверки

Чтобы изменить сведения о проверке:

1. Перейдите в раздел **Проверки соответствия** → **Проверки**. Система отобразит сведения об имеющихся проверках.
2. Нажмите на строку проверки в списке. Система отобразит страницу с карточкой этой проверки с подробной информацией о ней.
3. Перейдите на вкладку **Детали**.
4. Нажмите на кнопку **Изменить** (✎) рядом с полем **Проверка**. Поле станет доступным для редактирования.
5. Нажмите на стрелку ⌄.
6. Выберите новое значение проверки из выпадающего списка.
7. Нажмите на кнопку **Сохранить**.

Изменение устройства

Чтобы изменить сведения об устройстве:

1. Перейдите в раздел **Проверки соответствия** → **Проверки**. Система отобразит сведения об имеющихся проверках.
2. Нажмите на строку проверки в списке. Система отобразит страницу с карточкой этой проверки с подробной информацией о ней.
3. Перейдите на вкладку **Устройство**.
4. Нажмите на кнопку **Изменить** (✎) рядом с полем **Название**. Поле станет доступным для редактирования.
5. Нажмите на стрелку ⌄.
6. Выберите другое устройство из выпадающего списка.
7. Нажмите на кнопку **Сохранить**.

Удаление проверки соответствия

Чтобы удалить проверку:

1. Перейдите в раздел **Проверки соответствия** → **Проверки**. Система отобразит сведения об имеющихся проверках.
2. Нажмите на строку проверки в списке. Система отобразит страницу с карточкой этой проверки с подробной информацией о ней.
3. Нажмите на кнопку **Действия** (⋮) в карточке проверки.
4. Выберите команду **Удалить**.
5. Подтвердите удаление. Проверка будет удалена.

База проверок

Данный раздел описывает процесс работы с базой проверок соответствия в системе. Работа осуществляется в разделе **Проверки соответствия** → **База проверок** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют

> соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О базе проверок соответствия

База проверок соответствия представляет собой массив записей с информацией о проверках, включая ссылки и описание. При сканировании каждая проверка выполняется по очереди, а результат проверки фиксируется. Система извлекает сведения о результатах проверки из базы данных проверок, что исключает дублирование данных.

Файл, содержащий данные для проверки соответствия, можно обновить с помощью менеджера обновлений.

Интерфейс раздела

Панель инструментов включает в себя следующие компоненты:

- Поле **Поиск** предназначено для быстрого поиска записей базы проверок в отображаемом списке по идентификатору.
- Кнопка **Удалить** позволяет удалить запись базы проверок. Кнопка становится доступной после выбора хотя бы одной записи из списка.

В рабочей области отображается таблица имеющихся записей базы проверок. При выборе конкретной записи на экране открывается карточка с детальной информацией о ней.

Таблица содержит следующие столбцы:

- **Уровень критичности** — уровень критичности проверки. Доступные уровни:
  - **Высокий**;
  - **Средний**;
  - **Низкий**.
- **ID** — уникальный идентификационный код проверки.
- **Название** — название проверки.
- **Стандарты** — названия стандартов, в рамках которых проводилась проверка.
- **Теги** — теги, позволяющие классифицировать проверку.
- **Категория** — категория проверки.

При работе с таблицей объектов доступны следующие операции:

- Поиск объектов по полю **ID**.
- Сортировка объектов по полям **Уровень критичности**, **ID**, **Категория** и **Название**.
- Настройка отображения таблицы.

При выборе конкретной записи в правой части рабочей области отображается ее карточка с детальной информацией. Карточка записи содержит следующие вкладки:

- **Основное** — содержит основные сведения о проверке.

- **Детали** — содержит сведения об описании проверки, решении и результате проверки.

Работа с проверками соответствия

Доступные операции над проверками соответствия:

- Просмотр проверок соответствия
- Удаление проверок соответствия

Просмотр проверок соответствия

Чтобы просмотреть проверку соответствия:

1. Перейдите в раздел **Проверки соответствия** → **Проверки**. Система отобразит сведения об имеющихся проверках.
2. Нажмите на строку проверки в списке. Система отобразит в правой части экрана карточку этой проверки с подробной информацией о ней.

ID проверки отображается в верхней части карточки. Справа от названия расположена кнопка ⋮, по которой открывается выпадающее меню с действиями, доступными для выбранного объекта.

В карточке отображаются следующие вкладки:

- **Основное** — содержит основные сведения о проверке.
- **Детали** — содержит сведения об описании проверки, решении и результате проверки.

Вкладка Основное

На вкладке **Основное** отображаются основные сведения о проверке.

Вкладка содержит следующие поля:

- **ID проверки** — уникальный идентификационный код проверки.
- **Название** — название проверки.
- **Категория** — категория проверки.
- **Стандарт** — список стандартов, к которым относится проверка. Список представлен в виде таблицы со столбцом **Значения**, в котором отображаются наименования стандартов.
- **Уровень критичности** — уровень критичности проверки.
- **Теги** — список тегов проверки. Список представлен в виде таблицы со столбцом **Значения**, в котором отображаются наименования тегов проверки.

  Над таблицами вкладки доступны следующие операции:

  - Кнопка **Добавить** (＋) позволяет добавить новую запись.
  - При наведении курсора на строку записи отображается кнопка **Действия** (⋮). При нажатии на нее становятся доступны следующие команды:
    - **Изменить** — позволяет отредактировать запись.
    - **Удалить** — позволяет удалить запись.

- Сортировка записей.
- Вы можете развернуть таблицу с помощью кнопки ⌄ или свернуть с помощью кнопки ⌃.

Вкладка Детали

На вкладке **Детали** отображаются сведения об описании проверки, решении и результате проверки.

Вкладка содержит следующие поля:

- **Описание** — описание проверки.
- **Решение** — решение по проверке.

Удаление проверок соответствия

Чтобы удалить проверку:

1. Перейдите в раздел **Проверки соответствия** → **Проверки**. Система отобразит сведения об имеющихся проверках.
2. Нажмите на строку проверки в списке. Система отобразит страницу с карточкой этой проверки с подробной информацией о ней.
3. Выберите опцию **Удалить** в выпадающем меню кнопки действий ⋮ в верхней части карточки. Отобразится окно подтверждения удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранную проверку и отобразит соответствующее уведомление. Удаленная проверка будет исключена из списка раздела.

Вы также можете удалить сразу несколько проверок. Для этого:

1. Перейдите в раздел **Проверки соответствия** → **База проверок**. Система отобразит сведения об имеющихся проверках.
2. Выберите из списка проверки, которые необходимо удалить, с помощью флажка слева от названия проверки.
3. Нажмите на кнопку **Удалить** на панели инструментов. Система удалит выбранные проверки. Удаленные проверки будут исключены из списка раздела.

Управление уязвимостями

В данном разделе описана работа с уязвимостями, обнаруженными системой при сканировании оборудования, а также с базой данных, содержащей сведения об известных уязвимостях.

Содержание раздела:

- Уязвимости
- Задачи по устранению уязвимостей
- Изменение настроек интеграции с Jira
- База уязвимостей

Уязвимости

Данный раздел описывает процесс работы с уязвимостями, обнаруженными системой при сканировании оборудования.

Работа осуществляется в разделе **Управление уязвимостями** → **Уязвимости** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Об уязвимостях

Уязвимости в системе представляют собой сущности, которые содержат все сведения о собранных сканерами и добавленных вручную актуальных уязвимостях.

Интерфейс раздела

Панель инструментов включает в себя следующие компоненты:

- Кнопка **Добавить** позволяет добавить уязвимость в систему.
- Поле **Поиск** предназначено для быстрого поиска проверок уязвимостей в отображаемом списке по названию.
- Кнопка **Удалить** позволяет удалить уязвимость. Кнопка становится доступной после выбора хотя бы одной уязвимости из списка.

В рабочей области отображается таблица имеющихся уязвимостей. Таблица содержит следующие столбцы:

- **Название** — название уязвимости.
- **Уровень** — уровень критичности уязвимости. Доступные уровни:
  - **Критический**;
  - **Высокий**;
  - **Средний**;
  - **Низкий**;
  - **Информационный**.
- **Рейтинг** — рейтинг уязвимости.
- **Статус** — статус уязвимости.
- **Организация** — организация, связанная с уязвимостью.
- **Источник** — источник уязвимости.
- **Тип источника** — тип источника уязвимости.
- **Профиль** — профиль уязвимости.
- **Дата обнаружения** — дата первого обнаружения уязвимости на устройстве. Дата обнаружения не

изменяется на всем протяжении существования уязвимости на устройстве.

- **Дата открытия** — дата открытия уязвимости в системе. Дата открытия устанавливается, когда уязвимость впервые импортирована в систему или впервые связана с устройством, а также обновляется при каждом переводе уязвимости из любого другого статуса в статус **Открыта**.
- **Дата закрытия** — дата закрытия уязвимости в системе. Дата закрытия обновляется каждый раз при переводе уязвимости из любого другого статуса в статус **Закрыта**.
- **Дата последнего обнаружения** — показывает, когда в последний раз уязвимость была обнаружена на устройстве.
- **Дата изменения** — показывает, когда в последний раз уязвимость была изменена вручную или в результате импорта.

При работе с таблицей уязвимостей доступны следующие операции:

- Поиск уязвимостей по полю **Название**.
- Сортировка уязвимостей по полям **Название**, **Уровень**, **Рейтинг**, **Статус**, **Источник**, **Тип источника**, **Дата изменения**, **Дата закрытия**, **Дата обнаружения**, **Дата последнего обнаружения** и **Дата изменения**.
- Пользовательская фильтрация объектов по полям.
- Настройка отображения таблицы.

При выборе конкретной записи в правой части рабочей области отображается ее карточка с детальной информацией. Карточка записи содержит следующие вкладки:

- **Основное** — основные сведения об уязвимости.
- **Детали** — подробные сведения об уязвимости.
- **Устройство** — информация об устройстве, с которым связана уязвимость.
- **БДУ ФСТЭК** — сведения об уязвимости из базы данных уязвимостей ФСТЭК.
- **Уязвимое ПО** — информация об уязвимом ПО.
- **Устранение** — задачи по устранению уязвимости.

Работа с уязвимостями

Доступные операции над уязвимостями:

- Создание уязвимости
- Просмотр уязвимости
- Удаление уязвимости

Создание уязвимости

Чтобы создать уязвимость:

1. Перейдите в раздел **Управление уязвимостями** → **Уязвимости**. Система отобразит сведения об имеющихся уязвимостях.
2. Нажмите на кнопку **Добавить**. Отобразится окно создания уязвимости.

3. Выберите из выпадающего списка идентификатор организации, связанной с уязвимостью.
4. Введите название уязвимости.
5. Выберите из выпадающего списка идентификатор уязвимости.
6. Введите рейтинг уязвимости.
7. Введите уровень критичности уязвимости.
8. Выберите из выпадающего списка статус уязвимости.
9. Выберите из выпадающего списка источник данных уязвимости.
10. Выберите из выпадающего списка тип источника данных уязвимости.
11. Введите профиль сканирования, использовавшийся для обнаружения уязвимости.
12. Введите результат автоматического сканирования.
13. Задайте даты:
    - дату обнаружения уязвимости;
    - дату открытия уязвимости;
    - дату закрытия уязвимости;
    - дату последнего обнаружения уязвимости;
    - дату устранения уязвимости.

    Дату можно ввести вручную или задать с помощью панели выбора даты и времени.
14. Введите порт/протокол, по которому эксплуатируется уязвимость.
15. Выберите из выпадающего списка имя устройства, на котором была обнаружена уязвимость.
16. Выберите из выпадающего списка название ПО, содержащего данную уязвимость.
17. Нажмите на кнопку **Добавить**. Система создаст уязвимость и отобразит соответствующее уведомление. Новая уязвимость отобразится в списке раздела.

Просмотр уязвимости

Чтобы просмотреть сведения об уязвимости:

1. Перейдите в раздел **Управление уязвимостями** → **Уязвимости**. Система отобразит сведения об имеющихся уязвимостях.
2. Нажмите на строку уязвимости в списке. Система отобразит в правой части экрана карточку этой уязвимости с подробной информацией о ней. В карточке уязвимости доступны следующие вкладки:
    - **Основное** — основные сведения об уязвимости.
    - **Детали** — подробные сведения об уязвимости.
    - **Устройство** — информация об устройстве, с которым связана уязвимость.
    - **БДУ ФСТЭК** — сведения об уязвимости из базы данных уязвимостей ФСТЭК.
    - **Уязвимое ПО** — информация об уязвимом ПО.
    - **Устранение** — задачи по устранению уязвимости.

Вкладка Основное

На вкладке **Основное** отображаются основные сведения об уязвимости:

Вкладка содержит следующие поля:

- **Идентификатор организации** — уникальный идентификационный код организации, связанной с уязвимостью.
- **Название** — название уязвимости.
- **Рейтинг** — рейтинг уязвимости.
- **Уровень критичности** — уровень критичности уязвимости.
- **Статус** — статус уязвимости.
- **Тип источника данных** — тип источника данных уязвимости.
- **Источник данных** — источник данных уязвимости.
- **Профиль сканирования** — профиль сканирования уязвимости.
- **Дата обнаружения** — дата первого обнаружения уязвимости на устройстве. Дата обнаружения не изменяется на всем протяжении существования уязвимости на устройстве.
- **Дата открытия** — дата открытия уязвимости на устройстве. Дата открытия фиксируется при первоначальном импорте уязвимости в систему или при первой привязке к устройству, а также обновляется при каждом переводе уязвимости из любого другого статуса в статус **Открыта**.
- **Дата закрытия** — дата закрытия уязвимости в системе. Дата закрытия обновляется каждый раз при переводе уязвимости из любого другого статуса в статус **Закрыта**.
- **Дата последнего обнаружения** — дата последнего обнаружения уязвимости на устройстве.
- **Дата устранения** — дата устранения уязвимости.

Вкладка Детали

На вкладке **Детали** отображаются подробные сведения об уязвимости.

Вкладка содержит следующие поля:

- **Уязвимость** — название уязвимости. При нажатии на значение этого поля отобразится карточка уязвимости в базе данных объекта.
- **Описание** — описание уязвимости.
- **Решение** — решение по уязвимости.
- **Наличие эксплойта** — сведения о наличии эксплойта для уязвимости.
- **Трендовая** — информация, является ли уязвимость особо популярной.
- **CVE** — таблица со сведениями об уникальных идентификаторах, присваиваемых уязвимостям в рамках системы Common Vulnerabilities and Exposures (CVE). Таблица содержит следующие столбцы:
  - **CVE** — идентификатор.
  - **Ссылка** — ссылка на источник.
  - **Дата публикации** — дата публикации уязвимости.

- **Дата изменения** — дата изменения уязвимости.

- **CVSS** — раздел с оценками степени опасности уязвимости по шкале Common Vulnerability Scoring System (CVSS). В разделе доступны поля:
  - **Оценка CVSS** — оценка степени опасности уязвимости.
  - **Вектор CVSS** — описание серьёзности уязвимости через параметры её эксплуатации.
- **CVSS V2** — раздел с оценками степени опасности уязвимости по шкале Common Vulnerability Scoring System (CVSS V2). В разделе доступны поля:
  - **Оценка CVSS V2** — оценка уязвимости по шкале CVSS V2.
  - **Базовая оценка CVSS V2** — базовая оценка уязвимости по шкале CVSS V2.
  - **Временная оценка CVSS V2** — временная оценка уязвимости по шкале CVSS V2.
  - **Вектор CVSS V2** — вектор уязвимости по шкале CVSS V2.
  - **Базовый вектор CVSS V2** — базовый вектор уязвимости по шкале CVSS V2.
- **CVSS V3** — раздел с оценками степени опасности уязвимости по шкале Common Vulnerability Scoring System (CVSS V3). В разделе доступны поля:
  - **Оценка CVSS V3** — оценка уязвимости по шкале CVSS V3.
  - **Базовая оценка CVSS V3** — базовая оценка уязвимости по шкале CVSS V3.
  - **Временная оценка CVSS V3** — временная оценка уязвимости по шкале CVSS V3.
  - **Вектор CVSS V3** — вектор уязвимости по шкале CVSS V3.
  - **Базовый вектор CVSS V3** — базовый вектор уязвимости по шкале CVSS V3.
- **CVSS V4** — раздел с оценками степени опасности уязвимости по шкале Common Vulnerability Scoring System (CVSS V4). В разделе доступны поля:
  - **Оценка CVSS V4** — оценка уязвимости по шкале CVSS V4.
  - **Вектор CVSS V4** — вектор уязвимости по шкале CVSS V4.
- **Ссылки** — таблица со сведениями о ссылках на источники информации об уязвимости. Таблица содержит следующие столбцы:
  - **Источник** — название источника.
  - **Ссылка** — ссылка на источник.
- **Порт/Протокол** — порт или протокол, по которому эксплуатируется уязвимость.
- **Результат сканера** — результат автоматического сканирования.

Вкладка Устройство

На вкладке **Устройство** отображаются сведения об устройстве, с которым связана уязвимость.

Вкладка содержит следующие поля:

- **Имя устройства** — имя устройства, на котором проводилось сканирование.
- **Критичность** — уровень критичности проверки.
- **Статус** — статус проверки.
- **Домен / Рабочая группа** — название домена или рабочей группы, в которых производилась

проверка.

- **Операционная система** — сведения об операционной системе на сканируемом устройстве.
- **Тип узла** — тип сканируемого устройства.
- **Администратор безопасности** — имя администратора безопасности, контролирующего сканируемое устройство.
- **Сетевые интерфейсы** — таблица содержит сведения о сетевых интерфейсах сканируемого устройства. Сведения в таблице можно отсортировать по полю **IP-адрес**. Вы можете развернуть таблицу с помощью кнопки ⌄ или свернуть с помощью кнопки ⌃.

  Таблица содержит следующие столбцы:

  - **IP-адрес** — IP-адрес интерфейса в сети.
  - **Маска сети** — диапазон IP-адресов, принадлежащих той же подсети, что и интерфейс.
  - **MAC-адрес** — физический (аппаратный) адрес сетевого интерфейса, уникальный для устройства.
- **Группы ИТ-активов** — столбец **Название** раздела содержит сведения о группах ИТ-активов, связанных с устройством.

Вкладка БДУ ФСТЭК

На вкладке **БДУ ФСТЭК** отображаются сведения об уязвимости из базы данных уязвимостей ФСТЭК.

Вкладка содержит следующие поля:

- **Идентификатор** — идентификатор уязвимости.
- **Название** — название уязвимости.
- **Описание** — описание уязвимости.
- **Рейтинг ФСТЭК** — рейтинг ФСТЭК для уязвимости.
- **Критичность ФСТЭК** — критичность ФСТЭК для уязвимости.
- **Ссылка** — ссылка на официальное описание уязвимости.
- **Класс уязвимости** — класс, к которому относится уязвимость, исходя из её природы и механизма эксплуатации.
- **Подтверждение уязвимости** — статус проверки и достоверности информации об уязвимости.
- **Смягчение последствий** — информацию о рекомендуемых мерах, которые могут снизить негативное воздействие уязвимости.
- **Обновление** — дата и время последнего изменения информации об этой уязвимости в БДУ.
- **Верифицирована** — сведения о проверке и подтверждении уязвимости экспертами ФСТЭК.
- **Статус верификации** — статус проверки и подтверждения уязвимости экспертами ФСТЭК.
- **Дата публикации** — дата публикации уязвимости в БДУ.
- **Устранить до** — предписанный срок устранения обнаруженной уязвимости в соответствии с требованиями ФСТЭК.

Вкладка Уязвимое ПО

На вкладке **Уязвимое ПО** отображаются сведения о ПО, которое содержит данную уязвимость.

Вкладка содержит следующие поля:

- **ПО** — позволяет внести данные о ПО, которое содержит данную уязвимость, с помощью кнопки .
- **ПО** — список со сведениями о ПО, которое содержит данную уязвимость. Список представлен в виде таблицы.

  Таблица содержит следующие столбцы:

  - **Наименование ПО** — наименование программного обеспечения.
  - **Версия ПО** — версия программного обеспечения.

  При работе с таблицей доступны следующие операции:

  - Кнопка + позволяет добавить новую запись.
  - При наведении курсора на значение в каждом столбце отображается кнопка действий (⋮). При нажатии на нее становится доступна команда , которая позволяет отредактировать значение.
  - Вы можете развернуть таблицу с помощью кнопки ⌄ или свернуть с помощью кнопки ⌃ .

Вкладка Устранение

На вкладке **Устранение** отображаются задачи по устранению уязвимости.

Вкладка содержит следующие поля:

- **Задачи** — список задач по устранению уязвимости. Список представлен в виде таблицы.

  Таблица содержит следующие столбцы:

  - **Название задачи** — наименование задачи.
  - **Дата изменения** — дата последнего изменения задачи.

  При работе с таблицей доступны следующие операции:

  - В каждом столбце отображается кнопка действий (⋮). При нажатии на нее отображается выпадающее меню со следующими командами:
    - **Удалить** ( ) — позволяет удалить связанную задачу.
    - **Открыть** ( ) — позволяет открыть страницу связанной задачи в текущей вкладке.
    - **Открыть в новой вкладке** (↗) — позволяет открыть страницу связанной задачи в новой вкладке.
  - Вы можете развернуть таблицу с помощью кнопки ⌄ или свернуть с помощью кнопки ⌃ .
  - Сведения в таблице можно отсортировать.

Наличие эксплойта

Система может отображать информацию о наличии эксплойта для уязвимости. Эту информацию можно использовать при анализе и устранении уязвимости.

Для сбора информации об эксплойтах система использует оценки, полученные из баз данных систем Nexpose, Kaspersky Security Center, MaxPatrol SIEM, MaxPatrol 8, RedCheck, Qualys, OpenVAS, Vulners.com.

Поле **Наличие эксплойта** может принимать следующие значения:

- Нет данных;
- Широко распространен/Не требуется;
- Функциональный эксплойт;
- Демонстрационный эксплойт;
- Нет эксплойта.

Система автоматически обновляет значение поля **Наличие эксплойта** при создании или обновлении уязвимости.

Поиск совпадений статусов эксплойтов

Система ищет данные по метрике E (Exploit Code Maturity (степень зрелости доступных средств эксплуатации)) в оценке уязвимостей CVSS V3, а в случае их отсутствия — в оценке уязвимостей CVSS V2.

Если данные найдены, статус эксплойта в системе устанавливается в соответствии с таблицей ниже.

Статус эксплойта в системе	Данные по метрике E
Нет данных	E:ND/E:X
Широко распространен/Не требуется	E:H
Функциональный эксплойт	E:F
Демонстрационный эксплойт	E:POC
Нет эксплойта	E:U

Удаление уязвимости

Чтобы удалить уязвимость:

1. Перейдите в раздел **Управление уязвимостями** → **Уязвимости**. Система отобразит сведения об имеющихся уязвимостях.
2. Нажмите на строку уязвимости в списке. Система отобразит в правой части экрана карточку этой уязвимости с подробной информацией о ней.
3. Выберите опцию **Удалить** в выпадающем меню кнопки действий ⋮ в верхней части карточки.
4. Нажмите на кнопку **Удалить** в окне подтверждения. Система удалит выбранную уязвимость и отобразит соответствующее уведомление. Удаленная уязвимость будет исключена из списка раздела.

Вы также можете удалить сразу несколько уязвимостей. Для этого:

1. Перейдите в раздел **Управление уязвимостями** → **Уязвимости**. Система отобразит сведения об имеющихся уязвимостях.
2. В левом столбце таблицы установите флажки напротив тех уязвимостей, которые вы хотите удалить.
3. Нажмите на кнопку **Удалить** на панели инструментов. Система удалит выбранные уязвимости. Удаленные уязвимости будут исключены из списка раздела.

Статусы уязвимости

Смена статуса уязвимости

Уязвимостям, обнаруженным сканером безопасности на оборудовании, можно присвоить статусы.

Правила работы со статусами уязвимостей

- Уязвимости можно присвоить любой статус.
- Статус уязвимости определяется для конкретной уязвимости на оборудовании.
- Одна уникальная уязвимость может иметь разный статус на разном оборудовании.
- Изменить статус уязвимости на оборудовании можно с помощью политики управления уязвимостями или в интерфейсе вручную.

Статусы уязвимостей делятся на системные (**Открыта**, **Закрыта**) и пользовательские. Вы можете создать любое количество пользовательских статусов.

Системные статусы

Системные статусы доступны в системе по умолчанию. Для этих статусов можно отредактировать только описание.

- **Открыта**. Уязвимость актуальна. Присваивается автоматически после проведения сканирования и обнаружения этих уязвимостей на узле, или после открытия уязвимости.
- **Закрыта**. Проведены работы по устранению уязвимости. Присваивается автоматически при повторном сканировании, если уязвимость не обнаружена, вне зависимости от того, в каком статусе была уязвимость.

  При закрытии уязвимости система автоматически закрывает связанные с ней задачи. Если статус закрытой уязвимости вновь будет изменен, статусы связанных с ней и уже закрытых задач не изменятся.

Задачи по устранению уязвимостей

Данный раздел описывает процесс работы с задачами, направленными на устранение уязвимостей.

Работа осуществляется в разделе **Управление уязвимостями** → **Уязвимости** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют

соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О задачах

Задачи по устранению уязвимостей направлены на выявление и устранение потенциальных уязвимостей в системе, которые могут быть использованы злоумышленниками. Их выполнение позволяет повысить безопасность системы, обеспечить ее стабильность и предотвращать инциденты.

Работа с задачами

Доступные операции над задачами:

- Создание задачи
- Просмотр задачи
- Удаление задачи
- Экспорт сведений об уязвимости из задачи
- Создание, изменение и закрытие связанной задачи в Jira
- Просмотр связанной задачи в Jira

Создание задачи

Система позволяет создать в карточке уязвимости задачу по устранению уязвимости.

Чтобы создать задачу:

1. Перейдите в раздел **Управление уязвимостями** → **Уязвимости**. Система отобразит сведения об имеющихся уязвимостях.
2. Нажмите на строку уязвимости в списке. Система отобразит в правой части экрана карточку этой уязвимости с подробной информацией о ней.
3. Выберите опцию **Создать задачу** в выпадающем меню кнопки действий ⋮ в верхней части карточки. Система отобразит окно с параметрами запуска задачи.
4. Нажмите на кнопку **Создать задачу**. Система создаст задачу по устранению уязвимости и отобразит ее на вкладке **Устранение** в разделе **Задачи**.

Просмотр задачи

Система отображает на вкладке задачи критичность уязвимости. Остальные сведения об уязвимости доступны в документе в разделе **Документы**. Данные об организации, связанной с уязвимостью, автоматически передаются в связанную задачу Jira.

1. Перейдите в раздел **Управление уязвимостями** → **Уязвимости**. Система отобразит сведения об имеющихся уязвимостях.
2. Нажмите на строку уязвимости в списке. Система отобразит в правой части экрана карточку этой уязвимости с подробной информацией о ней.

3. Выберите вкладку **Устранение**. В разделе **Задачи** система отобразит список задач по устранению уязвимости.
4. Нажмите на кнопку действий (⋮) в строке задачи, которую необходимо просмотреть. Система отобразит список доступных команд.
5. Выберите команду **Открыть** (👁) или **Открыть в новой вкладке** (↗). Система отобразит на вкладке сведения о задаче.

Вкладка содержит следующие поля:

- **Название задачи** — название задачи по устранению уязвимости.
- **Исполнитель задачи** — исполнитель, назначенный для работы с задачей.
- **Ссылка на задачу в Jira** — ссылка на задачу в системе Jira. Отображается, если для текущей задачи была создана связанная задача Jira.
- **Приоритет задачи** — сведения о приоритете задачи.

  Если задача связана с задачей Jira, при изменении значения поля приоритет задачи Jira также изменится.

- **Идентификатор задачи в Jira** — сведения об идентификаторе, присвоенном связанной задаче в Jira.
- **Документы** — список документов, связанных с задачей. Список представлен в виде таблицы.

  Таблица содержит столбец **Значения** с названиями документов.

- **Статус** — текущий статус задачи. Возможные статусы:
  - **Новый**;
  - **В работе**;
  - **Закрыт**.

  Если текущая задача связана с задачей Jira, при изменении значения поля статус задачи Jira также изменится.

- **Идентификатор уязвимости** — идентификатор уязвимости, с которой связана задача.
- **Название уязвимости** — название уязвимости.
- **Источник уязвимости** — источник данных уязвимости.
- **Описание уязвимости** — сведения об описании уязвимости.
- **Описание решения для уязвимости** — сведения о решении, позволяющем устранить уязвимость.
- **Оценка CVSS V2** — оценка уязвимости по шкале CVSS V2.
- **Оценка CVSS V3** — оценка уязвимости по шкале CVSS V3.
- **Наличие эксплойта** - сведения о наличии эксплойта для уязвимости.
- **Критичность** — уровень критичности уязвимости.
- **Интерфейсы** — таблица содержит сведения о сетевых интерфейсах сканируемого устройства.

Таблица содержит следующие столбцы:

- **IP-адрес** — IP-адрес интерфейса в сети.
- **Маска сети** — диапазон IP-адресов, принадлежащих той же подсети, что и интерфейс.
- **MAC-адрес** — физический (аппаратный) адрес сетевого интерфейса, уникальный для устройства.

- **Ссылки** — таблица со сведениями о ссылках на источники информации об уязвимости.

  Таблица содержит следующие столбцы:

  - **Источник** — название источника.
  - **Ссылка** — ссылка на источник.
- **Дата обнаружения** - дата обнаружения уязвимости.
- **Комментарии к задаче** — таблица с комментариями к задаче по устранению уязвимости.

  В таблице отображаются комментарии, созданные в задаче Jira, связанной с текущей задачей. Создавать комментарии в текущей задаче нельзя. Если изменить или удалить комментарий в карточке Jira, комментарий в задаче карточки не изменится.

  Таблица содержит следующие столбцы:

  - **Комментарий** — содержание комментария.
  - **Автор** — автор комментария.
  - **Дата** - дата создания комментария.

    Над таблицами вкладки доступны следующие операции:

    - Кнопка + позволяет добавить новую запись.
    - При наведении курсора на значение в каждом столбце отображается кнопка , которая позволяет отредактировать значение.
    - При нажатии на кнопку ⋮ становятся доступны следующие команды:
      - **Изменить** — позволяет отредактировать запись.
      - **Удалить** — позволяет удалить запись.
    - Вы можете развернуть таблицу с помощью кнопки или свернуть с помощью кнопки .
- **Синхронизировать комментарии** — при нажатии на кнопку комментарии, добавленные к связанной задаче в Jira, синхронизируются с комментариями к задаче в системе.

Удаление задачи

Чтобы удалить задачу:

1. Перейдите в раздел **Управление уязвимостями** → **Уязвимости**. Система отобразит сведения об имеющихся уязвимостях.

2. Нажмите на строку уязвимости в списке. Система отобразит в правой части экрана карточку этой уязвимости с подробной информацией о ней.
3. Перейдите на вкладку **Устранение**.
4. Нажмите в разделе **Задачи** вкладки на кнопку действий (⋮) в строке задачи по устранению уязвимости. Система отобразит выпадающее меню доступных команд.
5. Выберите команду **Удалить** ( ).
6. Нажмите на кнопку **Удалить** в окне подтверждения. Система удалит выбранную задачу и отобразит соответствующее уведомление.

Экспорт сведений об уязвимости из задачи

Из задачи по устранению уязвимости можно экспортировать документ в формате XSLX со сведениями об уязвимости.

Чтобы экспортировать сведения об уязвимости:

1. Перейдите в раздел **Управление уязвимостями** → **Уязвимости**. Система отобразит сведения об имеющихся уязвимостях.
2. Нажмите на строку уязвимости в списке. Система отобразит в правой части экрана карточку этой уязвимости с подробной информацией о ней.
3. Перейдите на вкладку **Устранение**.
4. Нажмите на кнопку действий (⋮) в строке задачи по устранению уязвимости в разделе **Задачи**. Система отобразит выпадающее меню доступных команд.
5. Выберите команду **Открыть** ( ) или **Открыть в новой вкладке** (↗). Система отобразит на вкладке сведения о задаче.
6. Нажмите на строку документа в разделе **Документы**. Система откроет окно сохранения документа.
7. Укажите путь для сохранения документа и сохраните документ.

Создание, изменение и закрытие связанной задачи в Jira

В системе существует интеграция с внешней системой Jira. В Jira автоматически будет создаваться связанная задача, аналогичная задаче по устранению уязвимости в системе. При изменении приоритета или статуса одной из задач они также будут изменены в задаче, с которой она связана. При завершении работы над уязвимостью система будет автоматически закрывать обе задачи — и в системе, и в Jira.

Вы можете изменить настройки интеграции с Jira.

Просмотр связанной задачи в Jira

Вы можете перейти из карточки задачи в связанную задачу, созданную в системе Jira.

Чтобы перейти в задачу Jira:

1. Перейдите в раздел **Управление уязвимостями** → **Уязвимости**. Система отобразит сведения об имеющихся уязвимостях.

2. Нажмите на строку уязвимости в списке. Система отобразит в правой части экрана карточку этой уязвимости с подробной информацией о ней.
3. Перейдите на вкладку **Устранение**.
4. Нажмите на кнопку **Действия** (⋮) в строке задачи, связанной с уязвимостью, в разделе **Задачи**. Система отобразит список доступных команд.
5. Выберите команду **Открыть** (👁) или **Открыть в новой вкладке** (↗). Система отобразит на вкладке сведения о задаче.
6. Скопируйте ссылку из поля и вставьте ее в адресную строку браузера. Задача будет открыта в системе Jira.

Изменение настроек интеграции с Jira

Система интегрирована с внешней системой Jira. Это обеспечивает автоматическое создание связанной задачи в Jira, аналогичной задаче по устранению уязвимости в системе. Вы можете изменить существующие в системе настройки интеграции по умолчанию.

Предварительно убедитесь, что у вас имеется учетная запись Jira, и настройте в этой системе проекты, соответствующие вашей организации.

Настройки интеграции можно изменить, отредактировав параметры триггеров, контролирующих запуск определенных событий синхронизации при интеграции с Jira. Триггеры описаны в таблицах, представленных ниже.

- Триггеры типа Событие:

Название	Описание
[Jira VM EVO] Create Jira Task	Автоматическое создание задач в Jira при создании задачи системы
[Jira VM EVO] Sync Tasks CMS → Jira	Автоматическая синхронизация параметров задачи системы с параметрами задачи Jira
[Jira VM EVO] Close CMS Tasks	Автоматическое завершение работы над задачами системы и Jira при завершении работы над уязвимостью

- Триггеры типа Расписание:

Название	Описание
[Jira VM EVO] Sync Tasks Jira → CMS	Автоматическая синхронизация параметров задачи Jira с параметрами задачи системы
[Jira VM EVO] Sync Tasks Comments	Автоматическая синхронизация комментариев к задаче Jira с комментариями к задаче системы

Периодичность синхронизации параметров задачи Jira и комментариев к ней настраивается в поле **Cron-выражение** параметров триггера типа **Расписание**.

База уязвимостей

Данный раздел описывает процесс работы с базой, хранящей данные уникальных уязвимостей. С каждой уникальной уязвимостью связаны ее экземпляры — уязвимости, найденные на оборудовании.

Работа осуществляется в разделе **Управление уязвимостями** → **База уязвимостей** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

О базе уязвимостей

База уязвимостей представляет собой массив записей с данными уникальных уязвимостей. Она позволяет проводить анализ уязвимостей по различным критериям, расставлять приоритеты для их устранения, а также отслеживать их состояние:

- Общее состояние всех уязвимостей на оборудовании для каждой уникальной уязвимости: количество уязвимостей на оборудовании, суммарный рейтинг уникальной уязвимости.
- Текущее состояние всех открытых (необработанных) уязвимостей на оборудовании: количество открытых уязвимостей на оборудовании с распределением по уровням критичности.

Файл, содержащий уязвимости, можно обновить с помощью менеджера обновлений.

Интерфейс раздела

Панель инструментов включает в себя следующие компоненты:

- Поле **Поиск** предназначено для быстрого поиска записей базы уязвимостей в отображаемом списке по названию.
- Кнопка **Удалить** позволяет удалить запись базы уязвимостей. Кнопка становится доступной после выбора хотя бы одной записи из списка.

В рабочей области отображается таблица имеющихся записей базы уязвимостей. При выборе конкретной записи на экране открывается карточка с детальной информацией о ней.

Таблица содержит следующие столбцы:

- **Идентификатор** — идентификатор уязвимости.
- **Название** — название уязвимости.
- **Трендовая** — информация, является ли уязвимость особо популярной.
- **Наличие эксплойта** — сведения о наличии эксплойта для уязвимости.
- **Оценка CVSS** — оценка степени опасности уязвимости по шкале CVSS.
- **Вектор CVSS** — вектор уязвимости по шкале CVSS.
- **Оценка CVSS V2** — оценка степени опасности уязвимости по шкале CVSS V2.

- **Вектор CVSS V2** — вектор уязвимости по шкале CVSS V2.
- **Оценка CVSS V3** — оценка степени опасности уязвимости по шкале CVSS V3.
- **Вектор CVSS V3** — вектор уязвимости по шкале CVSS V3.
- **Оценка CVSS V4** — оценка степени опасности уязвимости по шкале CVSS V4.
- **Вектор CVSS V4** — вектор уязвимости по шкале CVSS V4.
- **Дата обновления** — дата обновления уязвимости.

При работе с таблицей объектов доступны следующие операции:

- Поиск объектов по полю **Название**.
- Сортировка объектов по полям **Название**, **Трендовая**, **Наличие эксплойта**, **Оценка CVSS**, **Оценка CVSS V2**, **Оценка CVSS V3**, **Оценка CVSS V4** и **Дата обновления**.
- Настройка отображения таблицы.

При выборе конкретной записи в правой части рабочей области отображается ее карточка с детальной информацией. Карточка записи содержит следующие вкладки:

- **Основное** — содержит основные сведения об уязвимости.
- **Оценки** — содержит сведения об оценках уязвимости
- **Решение** — содержит рекомендации по устранению уязвимости.
- **БД ФСТЭК** — содержит сведения об уязвимости из базы данных уязвимостей ФСТЭК.

Работа с уязвимостями

Доступные операции над уязвимостями

- Просмотр уязвимостей
- Удаление уязвимостей

Просмотр уязвимостей

Чтобы просмотреть уязвимость:

1. Перейдите в раздел **Управление уязвимостями** → **База уязвимостей**. Система отобразит сведения об имеющихся уязвимостях.
2. Нажмите на строку уязвимости в списке. Система отобразит в правой части экрана карточку этой уязвимости с подробной информацией о ней.

В верхней части карточки отображается ID уязвимости.

В карточке отображаются следующие вкладки:

- **Основное** — содержит основные сведения об уязвимости.
- **Оценки** — содержит сведения об оценках уязвимости
- **Решение** — содержит рекомендации по устранению уязвимости.
- **БДУ ФСТЭК** — содержит сведения об уязвимости из базы данных уязвимостей ФСТЭК.

Вкладка Основное

На вкладке **Основное** отображаются основные сведения об уязвимости.

Вкладка содержит следующие поля:

- **Идентификатор** — идентификатор уязвимости.
- **Название** — название уязвимости.
- **Описание** — описание уязвимости
- **CVE** — список со сведениями об уникальных идентификаторах, присвоенных уязвимости в рамках системы Common Vulnerabilities and Exposures (CVE). Список представлен в виде таблицы.

  Таблица содержит следующие столбцы:

  - **CVE** — наименование программного обеспечения.
  - **Ссылка** — версия программного обеспечения.
  - **Дата публикации** — дата публикации уязвимости.
  - **Дата изменения** — дата изменения уязвимости.
- **Трендовая** — информация, является ли уязвимость особо популярной.
- **Наличие эксплойта** — сведения о наличии эксплойта для уязвимости.
- **Наличие в каталоге CISA KEV** — уязвимость включена в Каталог известных эксплуатируемых уязвимостей (Known Exploited Vulnerabilities, KEV) Агентства кибербезопасности и инфраструктуры США (CISA).
- **Ссылки** — список со сведениями о ссылках на источники информации об уязвимости. Список представлен в виде таблицы.

  Таблица содержит следующие столбцы:

  - **Источник** — название источника.
  - **Ссылка** — ссылка на источник.
- **Эксплойты** — список со сведениями о наличии эксплойта для уязвимости. Список представлен в виде таблицы.

  Таблица содержит следующие столбцы:

  - **Источник эксплойта** — название источника.
  - **URL эксплойта** — ссылка на источник.

  Над таблицами вкладки доступны следующие операции:

  - Кнопка + позволяет добавить новую запись.
  - При наведении курсора на значение в каждом столбце отображается кнопка , которая позволяет отредактировать значение.
  - При нажатии на кнопку ⋮ становятся доступны следующие команды:
    - **Изменить** — позволяет отредактировать запись.
    - **Удалить** — позволяет удалить запись.

- Вы можете развернуть таблицу с помощью кнопки ⌄ или свернуть с помощью кнопки ⌃.

Вкладка Оценки

На вкладке **Оценки** отображаются сведения об оценках уязвимости.

Вкладка содержит следующие поля:

- **CVSS** — раздел с оценками степени опасности уязвимости по шкале Common Vulnerability Scoring System (CVSS). В разделе доступны поля:
  - **Оценка CVSS** — оценка степени опасности уязвимости.
  - **Вектор CVSS** — описание серьёзности уязвимости через параметры её эксплуатации.
- **CVSS V2** — раздел с оценками степени опасности уязвимости по шкале Common Vulnerability Scoring System (CVSS V2). В разделе доступны поля:
  - **Оценка CVSS V2** — оценка уязвимости по шкале CVSS V2.
  - **Базовая оценка CVSS V2** — базовая оценка уязвимости по шкале CVSS V2.
  - **Временная оценка CVSS V2** — временная оценка уязвимости по шкале CVSS V2.
  - **Вектор CVSS V2** — вектор уязвимости по шкале CVSS V2.
  - **Базовый вектор CVSS V2** — базовый вектор уязвимости по шкале CVSS V2.
- **CVSS V3** — раздел с оценками степени опасности уязвимости по шкале Common Vulnerability Scoring System (CVSS V3). В разделе доступны поля:
  - **Оценка CVSS V3** — оценка уязвимости по шкале CVSS V3.
  - **Базовая оценка CVSS V3** — базовая оценка уязвимости по шкале CVSS V3.
  - **Временная оценка CVSS V3** — временная оценка уязвимости по шкале CVSS V3.
  - **Вектор CVSS V3** — вектор уязвимости по шкале CVSS V3.
  - **Базовый вектор CVSS V3** — базовый вектор уязвимости по шкале CVSS V3.
- **CVSS V4** — раздел с оценками степени опасности уязвимости по шкале Common Vulnerability Scoring System (CVSS V4). В разделе доступны поля:
  - **Оценка CVSS V4** — оценка уязвимости по шкале CVSS V4.
  - **Вектор CVSS V4** — вектор уязвимости по шкале CVSS V4.
- **EPSS** — вероятность эксплуатации уязвимости в ближайшие 30 дней (Exploit Prediction Scoring System).

Вкладка Решение

На вкладке **Решение** отображаются рекомендации по устранению уязвимости.

Вкладка содержит следующие поля:

- **CVE** — список со сведениями об уникальном идентификаторе уязвимости, для которого предлагается решение. Список представлен в виде таблицы с полем **CVE**.
- **Ссылки** — список со ссылками на источники информации об уязвимости, для которой предлагается решение. Список представлен в виде таблицы с полем **Ссылка**.

Над таблицами вкладки доступны следующие операции:

- Кнопка + позволяет добавить новую запись.
- При наведении курсора на значение в каждом столбце отображается кнопка , которая позволяет отредактировать значение.
- При нажатии на кнопку ⋮ становятся доступны следующие команды:
  - **Изменить** — позволяет отредактировать запись.
  - **Удалить** — позволяет удалить запись.
- Вы можете развернуть таблицу с помощью кнопки ⌄ или свернуть с помощью кнопки ⌃.

Вкладка БДУ ФСТЭК

На вкладке **БДУ ФСТЭК** отображаются сведения об уязвимости из базы данных уязвимостей ФСТЭК.

Вкладка содержит следующие поля:

- **Идентификатор** — идентификатор уязвимости.
- **Название** — название уязвимости.
- **Описание** — описание уязвимости.
- **Ссылка** — ссылка на официальное описание уязвимости.
- **Класс уязвимости** — класс, к которому относится уязвимость, исходя из её природы и механизма эксплуатации.
- **Подтверждение уязвимости** — статус проверки и достоверности информации об уязвимости.
- **Дата публикации** — дата публикации уязвимости в БДУ.
- **Смягчение последствий** — информацию о рекомендуемых мерах, которые могут снизить негативное воздействие уязвимости.
- **Обновление** — дата и время последнего изменения информации об этой уязвимости в БДУ.
- **Верифицирована** — сведения о проверке и подтверждении уязвимости экспертами ФСТЭК.
- **Статус верификации** — статус проверки и подтверждения уязвимости экспертами ФСТЭК.

Удаление уязвимостей

Чтобы удалить одну или несколько уязвимостей:

1. Перейдите в раздел **Управление уязвимостями** → **База уязвимостей**. Система отобразит сведения об имеющихся уязвимостях.
2. В левом столбце таблицы установите флажки напротив тех уязвимостей, которые вы хотите удалить.
3. Нажмите на кнопку **Удалить** на панели инструментов. Система удалит выбранные уязвимости. Удаленные уязвимости будут исключены из списка раздела.

Менеджер обновлений

Менеджер обновлений позволяет:

- Обновить базу уязвимостей:
  - автоматически;
  - вручную с сервера обновлений;
  - вручную из архива базы уязвимостей.
- Обновить файл проверки соответствия для работы с задачами проверки соответствия.

Работа с менеджером обновлений осуществляется в разделе **Сканирование** → **Менеджер обновлений**.

Обновление базы уязвимостей

Обновление базы уязвимостей осуществляется в разделе **Сканирование** → **Менеджер обновлений**.

Непрерывная работоспособность ссылок на публичные эксплойты, содержащихся в базе уязвимостей, не гарантируется. Разработчики базы уязвимостей не контролируют содержимое или авторство сайтов, на которые ведут эти ссылки, и не несут ответственности за их содержимое, которое может быть изменено после публикации ссылок. Содержимое по таким ссылкам может быть в любое время удалено как автором, так и по решению правообладателя.

Разработчики базы уязвимостей прилагают усилия для своевременной и регулярной проверки актуальности ссылок на публичные эксплойты. Планируется пополнение базы данными с использованием постоянных ссылок.

Базу уязвимостей можно обновить несколькими способами:

- **Автоматически** — при подключении к серверу обновлений R-Vision согласно указанному расписанию.

  База уязвимостей обновляется каждые 24 часа, поддерживая, таким образом, актуальность сведений по уже покрываемым источникам.

- **Вручную с сервера обновлений** — при подключении к серверу обновлений R-Vision с помощью кнопки **Обновить**.
- **Вручную из архива базы уязвимостей** — при загрузке в систему архива базы уязвимостей в формате .vulns из указанного месторасположения.

В систему нельзя загрузить базу уязвимостей, выпущенную ранее, чем база, которая используется в настоящий момент.

При успешном импорте, распаковке и проверке файлов архива система начнет использовать загруженную версию базы уязвимостей. Информация о базе уязвимостей, которая используется в системе в настоящий момент, обновится.

В верхней части экрана система отображает информацию о базе уязвимостей, которая используется в системе в настоящий момент:

- дату и время, когда база была выпущена;
- дату и время, когда база была загружена в систему.

Автоматическое обновление базы уязвимостей

Настройки автоматического обновления выполняются в разделе **Параметры обновления**. Здесь также можно проверить подключение к серверу обновлений базы уязвимостей R-Vision. В нижней части раздела отображается информация о статусе автоматического обновления в системе (включено, выключено) и дате следующего автоматического обновления, если оно настроено.

Чтобы настроить автоматическое обновление базы уязвимостей, выполните следующие действия:

1. Предоставьте системе доступ к серверу по адресу `https://sec-hub.rvision.pro/union/feed.vulns` через порт 443 (протокол TCP).
2. Перейдите в раздел **Сканирование** → **Менеджер обновлений**.
3. В разделе **Параметры обновления** нажмите на кнопку **Настроить**. Система отобразит окно настройки параметров.
4. Установите флажок **Автоматическое обновление базы уязвимостей**.
5. Укажите параметры расписания:
   a. Периодичность запуска: по дням, по неделям, по месяцам.
   b. Режим повторов запуска:
      - Для режима **По дням** – интервал повтора в днях и времени начала.
      - Для режима **По неделям** – интервал повтора в неделях с указанием дней недели и времени начала.
      - Для режима **По месяцам** – интервал повтора в месяцах с указанием дней месяца и времени начала.
6. При необходимости установите флажок **Использовать прокси-сервер** и укажите данные для подключения к серверу.
7. Нажмите на кнопку **Подтвердить**. Система будет обновлять базу уязвимостей согласно указанным настройкам.

Обновление базы уязвимостей вручную

Вы можете обновить базу уязвимостей, загрузив в систему архив базы уязвимостей с расширением .vulns вручную.

Чтобы обновить базу, загрузив файл вручную:

1. Перейдите в раздел **Сканирование** → **Менеджер обновлений**.
2. Нажмите на кнопку в верхней части экрана. Система отобразит окно загрузки, в котором нужно выбрать файл с расширением .vulns, содержащий архив базы уязвимостей. Размер файла не может превышать 2 ГБ.

3. Нажмите на кнопку **Подтвердить**. Файл будет загружен в систему.

Обновление базы уязвимостей вручную с сервера обновлений

Чтобы вручную запустить обновление базы уязвимостей с сервера обновлений:

1. Перейдите в раздел **Сканирование → Менеджер обновлений**.
2. Нажмите на кнопку **Обновить** в верхней части экрана. Обновление будет запущено.

Если необходимо отменить обновление, нажмите на кнопку **Отмена**.

Информация о процессе обновления

Во время выполнения обновления система отображает следующую информацию о процессе в разделе **Детали последнего обновления**:

- Дата и время запуска.
- **Источник** — сервер обновлений или файл.
- **Статус обновления** — сведения о статусе процесса обновления. Возможные значения:
  - **Не выполнялось**.
  - **Выполняется**.
  - **Ошибка**.
  - **Успешно**.
  - **Отменено**.
- **Подробнее** — раскрывающийся блок содержит поэтапные сведения о процессе установки обновления.

Проверка подключения к серверу обновлений базы уязвимостей

В разделе **Параметры обновления** можно проверить подключение к серверу обновлений базы уязвимостей R-Vision. Для этого нажмите на кнопку **Проверить подключение** (🔄).

В зависимости от результатов проверки система отобразит уведомление:

- **Доступен** — система подключена к серверу обновлений.
- **Недоступен** — не удалось подключиться к серверу обновлений.

Обновление файла проверки соответствия

При работе с задачами сканирования сети можно создать и настроить профиль сканирования типа **Проверка стандартов**. Задачи, запускаемые в рамках этого профиля, позволяют проверить соответствие системы установленным нормативным требованиям и стандартам безопасности.

Для выполнения проверки системы на соответствие стандартам безопасности необходимо выбрать требуемые элементы из списка доступных стандартов. Эти данные содержатся в специальном файле проверки соответствия, который можно обновить. Для обновления файла используйте менеджер обновлений.

Чтобы загрузить в систему файл проверки соответствия, выполните следующие действия:

1. Перейдите в раздел **Сканирование** → **Менеджер обновлений**.
2. Нажмите на кнопку в верхней части экрана. Система отобразит окно загрузки обновлений.
3. Нажмите на кнопку **Выбрать файл** для выбора файла с расширением .co, содержащего сведения проверки соответствия. Размер файла не может превышать 2 ГБ.
4. Нажмите на кнопку **Подтвердить**. Файл будет загружен в систему.

Использование коллекции Postman

Функции API системы представлены в коллекции (collection) Postman. Коллекция Postman доступна здесь.

Чтобы начать работать с коллекцией, воспользуйтесь приложением Postman или веб-версией, доступной при регистрации на сайте.

1. В Postman нажмите на кнопку **Import** на правой панели и импортируйте скачанную коллекцию.
2. Выберите вкладку **Public API v2 <version>** с коллекцией (где <version> - версия системы) и перейдите во вкладку **Variables** на средней панели.
3. Введите и заполните следующие глобальные переменные:
   - `{{url}}` — адрес системы;
   - `{{token}}` — ваш API-токен.

Опционально для тех или иных запросов вам потребуется заполнить локальные переменные. В этом случае вы также можете ввести их на вкладке **Variables**.

Чтобы заполнить тело запроса, откройте вкладку соответствующего запроса, затем перейдите на вкладку **Body**.

- В окне **Query** отобразится запрос.
- В окне **GRAPHQL Variables** отобразятся переменные в формате JSON.

Для отправки запроса нажмите на кнопку **Send**. Результат запроса отобразится в поле **Response**.

Экспертиза

В данном разделе описаны инструменты и методы работы с элементами экспертизы в системе, включая схемы RObject и язык преобразования данных VRL.

Содержание раздела:

- Работа с элементами экспертизы
- Правила корреляции
  - Построение правила корреляции в конструкторе
  - Императивные правила
  - Декларативные правила
- Правила нормализации
- Правила агрегации
- Схемы активных списков
- Схемы дашбордов
- Схемы шаблонов отчетов
- Таблицы обогащения
- Шаблоны поиска
- Глобальные функции
- Схемы доменов
  - Структура схемы домена
  - Описания интерфейса
  - Типы данных в схемах доменов
- Скрипты
  - API в скриптах
- Плейбуки
  - Модули плейбуков
  - Специальные переменные в плейбуках
    - Переменная results
- Схемы конфигурации
- Схемы расчетов
  - Наследование в схемах расчетов
- Тестирование правил экспертизы с помощью языка VRL

Работа с элементами экспертизы

Данный раздел описывает процесс работы с элементами экспертизы в системе. Работа

осуществляется в разделе **Экспертиза** веб-интерфейса системы.

Раздел доступен администраторам, пользователям, которые имеют соответствующее разрешение в составе своих ролей, а также учетным записям со статусом **Суперадминистратор**.

Об элементах экспертизы

Экспертиза — модуль системы, позволяющий создавать объекты на языке RObject для специализированных задач. RObject представляет собой синтаксическую надстройку над YAML. Объекты, создаваемые в экспертизе, называются **элементами экспертизы**.

Система позволяет работать со следующими элементами экспертизы:

- Правила корреляции
- Правила нормализации
- Правила агрегации
- Схемы активных списков
- Схемы дашбордов
- Схемы шаблонов отчетов
- Таблицы обогащения
- Шаблоны поиска
- Глобальные функции
- Схемы расчетов

В системе доступны элементы экспертизы двух типов:

- **Системные элементы экспертизы** — создаются системой автоматически. Они недоступны для переноса между каталогами, изменения, удаления или экспорта.
- **Пользовательские элементы экспертизы** — создаются вручную.

Интерфейс раздела

Панель инструментов раздела включает в себя следующие компоненты:

- Кнопка **Создать** (+) позволяет создать новый элемент экспертизы. По этой кнопке открывается выпадающий список для выбора типа создаваемого элемента.
- Кнопка **Удалить** ( ) позволяет удалять выбранные элементы.
- Кнопка **Переместить в каталог** ( ) позволяет переносить выбранные из списка элементы экспертизы в другой каталог.
- Кнопки **Экспортировать** ( ) и **Импортировать** ( ) позволяют выгружать опубликованные элементы экспертизы из системы и загружать в систему элементы извне.
- Переключатель **Каталоги** позволяет показать или скрыть в левой части рабочей области панель с деревом каталогов.

- Поле **Поиск** предназначено для быстрого поиска элементов экспертизы в отображаемом списке.
- Кнопка **Фильтр** ( ) позволяет настроить критерии фильтрации элементов экспертизы. В параметрах фильтрации доступны следующие опции:
  - **Тип элемента** — фильтрует элементы экспертизы по их типу.
  - **Тег** — фильтрует элементы по тегам, указанным в поле `tags` элемента, что позволяет отобрать элементы, относящиеся к определенным категориям.
  - **Источник данных** — фильтрует элементы по значению, указанному в поле `data_source` элемента, что позволяет отобрать элементы, совместимые с определенным источником данных.
  - **Статус** — фильтрует элементы по их текущему состоянию, например, включен или выключен.
  - **Уровень угрозы** — фильтрует элементы по уровню угрозы, например, критический, высокий, средний или низкий.
  - **Тип создания** — фильтрует элементы по типу их создания, например, пользовательский или системный.

Панель **Каталоги** содержит корневые каталоги и вложенные каталоги, относящиеся к ним. При нажатии на кнопку ⋮ в верхнем правом углу панели открывается выпадающее меню с кнопкой **Создать каталог** для добавления нового корневого каталога.

При наведении курсора на название вложенного каталога в строке появляется кнопка действий (⋮). При нажатии на нее открывается выпадающее меню с действиями, доступными для выбранного каталога: создание нового вложенного каталога, перемещение, переименование и удаление.

При выборе каталога система отображает список элементов экспертизы в этом каталоге. Список представлен в виде таблицы со следующими столбцами:

- **Тип элемента** — тип элемента экспертизы.
- **ID** — уникальный идентификационный код, который автоматически присваивается системой при создании элемента.
- **Название** — имя элемента экспертизы, используемое для его идентификации в системе.
- **Версия** — версия элемента экспертизы, представленная в формате SemVer v2.0. Неопубликованные элементы экспертизы (черновики) имеют значение **Черновик**.
- **Теги** — пользовательские теги, присвоенные данному элементу экспертизы.
- **Источник данных** — список совместимых с элементом экспертизы источников данных, присвоенных элементу в поле `data_source`.
- **Дата изменения** — дата и время последнего изменения элемента экспертизы.
- **Уровень угрозы** — уровень угрозы, ассоциированный с элементом экспертизы (**Низкий**, **Средний**, **Высокий**, **Критический** или **Не указан**).
- **Статус** — текущий статус элемента экспертизы (включен/выключен).

При работе с таблицей элементов экспертизы доступны следующие операции:

- Поиск элементов экспертизы. Поиск осуществляется по полям **ID** и **Название**, а также по коду

текущих версий элементов.

- Фильтрация элементов экспертизы по полям **Тип элемента**, **Тег**, **Источник данных**, **Статус**, **Уровень угрозы**, **Тип создания**:
  - Теги и источники данных вводятся вручную и применяются по клавише ENTER.
  - Тип элемента, статус, уровень угрозы и тип создания выбираются из выпадающих списков.

Для удобства работы с разделом заданные условия фильтрации отображаются над списком элементов. Условие фильтрации можно сбросить по кнопке ✕ справа от него. Чтобы сбросить сразу все условия, можно использовать кнопку **Сбросить все** (✕) справа от списка условий.

- Сортировка элементов экспертизы по полям **Название** и **Дата изменения**.
- Настройка отображения таблицы.

При выборе конкретного элемента экспертизы в правой части рабочей области отображается карточка с его детальной информацией. Карточка содержит следующие вкладки:

- **Информация** — общие сведения об элементе.
- **Связанные элементы** — элементы экспертизы, связанные с текущим элементом.
- **Конвейеры** — конвейеры, на которых используется элемент (только для правил).

Работа с каталогами

Элементы экспертизы хранятся в каталогах и подкаталогах. В левой части рабочей области интерфейса представлена иерархическая структура каталогов. Корневые каталоги содержат общий перечень элементов экспертизы, включая элементы из подкаталогов. При выборе конкретного каталога в древовидной структуре, содержащиеся в нем элементы экспертизы отображаются в таблице в центральной части рабочей области.

Доступные операции над каталогами:

- Создание каталога
- Переименование каталога
- Перемещение каталога
- Удаление каталога

Создание каталога

Вы можете создавать корневые или вложенные каталоги в разделе **Экспертиза**.

Чтобы создать корневой каталог:

1. Перейдите в раздел **Экспертиза**. В левой части рабочей области находится панель **Каталоги**, которая содержит перечень доступных в системе каталогов.
2. Нажмите на кнопку ⋮ справа от названия панели и выберите опцию **Создать каталог**. Отобразится окно создания каталога.

3. Введите название каталога и нажмите на кнопку **Создать**. Система создаст каталог и отобразит соответствующее уведомление. Новый каталог появится в списке.

Имена корневых каталогов должны быть уникальными с учетом регистра.

Чтобы создать вложенный каталог:

1. Выберите в дереве каталог, в котором требуется создать вложенный каталог.
2. Нажмите на появившуюся кнопку действий (⋮) напротив названия каталога и выберите опцию **Создать каталог**. Отобразится окно создания каталога.
3. Введите название каталога и нажмите на кнопку **Создать**. Система создаст каталог и отобразит соответствующее уведомление. Новый каталог появится в дереве внутри выбранного каталога.

Имена вложенных каталогов в пределах родительского каталога должны быть уникальными с учетом регистра.

Переименование каталога

Переименовать можно только вложенные каталоги. Имена каталогов в пределах родительского каталога должны быть уникальными с учетом регистра.

Чтобы переименовать каталог:

1. Перейдите в раздел **Экспертиза**. В левой части рабочей области находится панель **Каталоги**, которая содержит перечень доступных в системе каталогов.
2. Нажмите на появившуюся кнопку действий (⋮) напротив названия каталога и выберите опцию **Переименовать**. Отобразится окно переименования каталога.
3. Измените название каталога и нажмите на кнопку **Переименовать**. Система переименует каталог и отобразит соответствующее уведомление. Каталог отобразится в дереве под новым именем.

Перемещение каталога

Переместить можно только вложенные каталоги. Имена каталогов в пределах родительского каталога должны быть уникальными с учетом регистра.

Чтобы переместить каталог:

1. Перейдите в раздел **Экспертиза**. В левой части рабочей области находится панель **Каталоги**, которая содержит перечень доступных в системе каталогов.
2. Нажмите на появившуюся кнопку действий (⋮) напротив названия каталога и выберите опцию **Переместить**. Отобразится окно перемещения каталога.
3. В окне перемещения каталога раскройте дерево каталогов и выберите новый родительский каталог для текущего каталога.
4. Нажмите на кнопку **Переместить**. Система переместит каталог со всеми вложенными каталогами и элементами в выбранное местоположение и отобразит соответствующее

уведомление.

Удаление каталога

Удалить можно только вложенные каталоги, которые не содержат пользовательских элементов экспертизы.

1. Перейдите в раздел **Экспертиза**. В левой части рабочей области находится панель **Каталоги**, которая содержит перечень доступных в системе каталогов.
2. Нажмите на появившуюся кнопку действий (⋮) напротив названия каталога и выберите опцию **Удалить**. Система отобразит окно подтверждения удаления.
3. Нажмите **Удалить**. Система удалит каталог и отобразит соответствующее уведомление. Удаленный каталог исчезнет из дерева.

Работа с элементами экспертизы

К элементам экспертизы относятся:

- Правила корреляции
- Правила нормализации
- Правила агрегации
- Схемы активных списков
- Схемы дашбордов
- Схемы шаблонов отчетов
- Таблицы обогащения
- Шаблоны поиска
- Глобальные функции
- Схемы расчетов

Доступные операции над элементами экспертизы:

- Версионирование элемента экспертизы
- Создание элемента экспертизы
- Изменение элемента экспертизы
- Работа в окне настройки
- Тестирование элемента экспертизы
- Просмотр элемента экспертизы
- Включение элемента экспертизы
- Выключение элемента экспертизы
- Обновление элемента экспертизы на конвейерах
- Перенос элемента экспертизы

- Создание копии элемента экспертизы
- Копирование кода элемента экспертизы
- Удаление элемента экспертизы
- Экспорт элементов экспертизы
- Импорт элементов экспертизы

Версионирование элемента экспертизы

Работа с элементом экспертизы зависит от типа его версии. Доступны следующие типы:

- **Опубликованная версия** — полноценный рабочий элемент экспертизы. При публикации элемента экспертизы система автоматически запускает тесты для валидации его конфигурации.

  Для опубликованной версии доступны все возможные операции над элементом экспертизы, в том числе включение и выключение. Включенные элементы экспертизы можно использовать в элементах конвейера.

  При первичной публикации элемента экспертизы требуется указать его версию в формате SemVer v2.0 в поле `version`. При повторной публикации элемента экспертизы необходимо вручную увеличивать его версию. Историю изменения версий можно просмотреть в окне просмотра конфигурации опубликованного элемента экспертизы.

- **Черновик** — предварительный вариант элемента экспертизы, не предназначенный для полноценного использования в системе. Черновик позволяет изменять конфигурацию элемента экспертизы и предварительно сохранять внесенные изменения без публикации элемента и без валидации его настроек.

  В отличие от опубликованной версии, черновик нельзя включить, а следовательно, он недоступен для использования в элементах конвейера. Также для черновика недоступна возможность экспорта конфигурации.

Создание элемента экспертизы

Чтобы создать элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка нужный тип элемента.

   Доступные типы элементов экспертизы:

   - Правила корреляции
   - Правила нормализации
   - Правила агрегации
   - Схемы активных списков
   - Схемы дашбордов
   - Схемы шаблонов отчетов

- Таблицы обогащения
- Шаблоны поиска
- Глобальные функции
- Схемы расчетов

Система отобразит окно создания элемента экспертизы выбранного типа.

При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля элемента экспертизы, чтобы определить логику его работы.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст элемент экспертизы и отобразит соответствующее уведомление. Новый элемент появится в списке.

Чтобы элемент экспертизы стал доступен для использования в системе, его необходимо включить.

Вы можете создать черновик элемента экспертизы без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик элемента и отобразит соответствующее уведомление. Новый черновик появится в списке элементов экспертизы.

Представленная инструкция применима к элементам экспертизы всех типов, кроме правил корреляции.

Декларативные правила корреляции можно создавать как через RObject-конфигурацию, так и с помощью специального конструктора. Подробности приведены в разделе Создание правила корреляции.

Изменение элемента экспертизы

Изменение доступно только для элементов экспертизы с типом создания **Пользовательский**.

Чтобы изменить элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку элемента экспертизы в списке. Система отобразит в правой части экрана карточку с подробной информацией об элементе.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек элемента экспертизы.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

4. Внесите требуемые изменения в конфигурацию элемента экспертизы.
5. Нажмите на кнопку **Обновить версию**. Система опубликует новую версию элемента с измененной конфигурацией и отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию элемента экспертизы в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки элемента экспертизы будет отображаться предупреждение о наличии неопубликованных изменений.

Представленная инструкция применима к элементам экспертизы всех типов, кроме правил корреляции.

Декларативные правила корреляции можно изменять как через RObject-конфигурацию, так и с помощью специального конструктора. Подробности приведены в разделе Изменение правила корреляции.

Работа в окне настройки

При создании или изменении элемента экспертизы в окне настройки его конфигурации доступны:

- горячие клавиши;
- запуск тестов.

Для удобной работы с кодом предусмотрен полноэкранный режим. Чтобы развернуть окно на всю страницу браузера, нажмите в правом верхнем углу окна на кнопку **На весь экран** (⤢).

Чтобы вернуть обычный режим отображения, нажмите в правом верхнем углу кнопку **Свернуть** (⤡).

Чтобы скопировать код элемента, не открывая редактор, нажмите на кнопку действий (⋮) в верхнем правом углу карточки элемента и выберите опцию **Скопировать код**.

В VRL-программе для доступа к данным события используется символ точки (`.`), а к метаданным события — символ процента (`%`). В корреляционных событиях метаданные включают данные исходных событий.

Использование горячих клавиш

При работе в окне настройки доступно использование следующих горячих клавиш:

Функция	Клавиши и их комбинации
Сместить табуляцию на шаг вправо	TAB
Сместить табуляцию на шаг влево	SHIFT+TAB
Вызвать подсказки кода Чтобы вставить выбранную подсказку, нажмите TAB или ENTER.	CTRL+SPACE
Искать по коду	CTRL+F
Включить или выключить поиск с учетом регистра	ALT+C
Включить или выключить поиск слова целиком	ALT+W
Включить или выключить поиск с регулярными выражениями	ALT+R
Выделить текущее слово.	CTRL+D Чтобы выделить следующее идентичное значение и добавить курсор после него, удерживая CTRL, нажмите D.
Добавить курсор на следующие строки	CTRL+ALT+↓
Добавить курсор на предыдущие строки	CTRL+ALT+↑
Выделить всю строку	CTRL+L
Выделить следующую строку	SHIFT+↓
Выделить предыдущую строку	SHIFT+↑
Выделить весь код	CTRL+A
Копировать выделенное в буфер обмена	CTRL+C
Вырезать выделенное	CTRL+X
Вставить из буфера обмена	CTRL+V
Переместить текущую строку вниз	ALT+↓
Переместить текущую строку вверх	ALT+↑
Дублировать текущую строку и переместить курсор вниз	SHIFT+ALT+↓
Дублировать текущую строку и оставить курсор на текущей строке	SHIFT+ALT+↑
Отменить последнее действие	CTRL+Z
Повторить отмененное действие	CTRL+Y

Тестирование элемента экспертизы

В окне настройки конфигурации вы можете запустить тесты из блока тестирования `tests`. Для этого нажмите кнопку **Запустить тесты** (▷) в левой нижней части окна.

Тесты также запускаются автоматически при создании и публикации элемента экспертизы.

Если в конфигурации элемента экспертизы имеются ошибки, их требуется предварительно исправить перед запуском тестирования или публикацией элемента. Публикация элемента с ошибками в конфигурации недоступна.

Если тесты в блоке `tests` заданы, то система запустит тестирование и отобразит в нижней части окна панель с подробным описанием результатов тестов.

Если в правиле корреляции настроена корреляция по полю с типом `DateTime`, например `timestamp`, то значение этого поля в блоке `tests` необходимо задавать в формате `t'<DateTime-значение>'`. Например: `t'15.05.2024 06:28:15'`. Это важно при корреляции на отсутствие событий.

В противном случае при запуске тестов система обработает поле в виде строки и корреляция по дате-времени будет работать некорректно.

При необходимости вы можете свернуть панель результатов тестирования по нажатию кнопки ⌄ в правой части ее заголовка. Чтобы отобразить свернутую панель, нажмите на кнопку ⌃ в правой части ее заголовка.

Обновление элемента экспертизы на конвейерах

Чтобы обновить версию правила нормализации, корреляции или агрегации в рамках одного или нескольких конвейеров:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку элемента экспертизы в списке. Система отобразит в правой части экрана карточку с подробной информацией об элементе.
3. Откройте в карточке вкладку **Конвейеры** и нажмите на кнопку **Обновить версию** в нижней части вкладки. Отобразится дерево коллекторов с конвейерами, на которых установлено правило.

Кнопка **Обновить версию** отображается, если правило установлено хотя бы на одном конвейере.

4. Выберите конвейеры, на которых необходимо обновить версию правила, установив флажки слева от их названий.
5. Нажмите на кнопку **Обновить версию**. Система обновит правило до последней версии на выбранных конвейерах.

Просмотр элемента экспертизы

Чтобы просмотреть элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку элемента экспертизы в списке. Система отобразит в правой части экрана карточку с подробной информацией об элементе.

В верхней части карточки отображается название элемента экспертизы.

Чтобы изменить ширину карточки, наведите курсор на ее левую границу. Курсор примет вид двунаправленной стрелки. Перетащите левую границу карточки в нужном направлении.

Если элемент экспертизы имеет неопубликованные изменения, то в верхней части карточки отображается предупреждение.

Карточка содержит название элемента, кнопку ⋮, кнопку **Просмотр** либо **Изменить**, а также следующие вкладки:

- **Информация** — общие сведения об элементе.
- **Связанные элементы** — элементы экспертизы, связанные с текущим элементом.
- **Конвейеры** — конвейеры, на которых используется элемент (только для правил).

Вкладка Информация

По умолчанию при выборе элемента экспертизы открывается вкладка **Информация**. На вкладке отображается информация о выбранном элементе экспертизы.

Вкладка содержит следующие параметры:

- **Уровень угрозы** — уровень угрозы, ассоциированный с элементом экспертизы (**Низкий**, **Средний**, **Высокий**, **Критический** или **Не указан**).
- **Название** — имя элемента экспертизы, используемое для его идентификации в системе.
- **ID** — уникальный идентификационный код элемента экспертизы, генерируемый автоматически при создании элемента.
- **Тип элемента** — тип элемента экспертизы.
- **Тип создания** — тип создания элемента экспертизы: системный или пользовательский.
- **Описание** — краткое описание функциональности элемента экспертизы.
- **ID тенанта** — уникальный идентификационный код тенанта, к которому относится элемент экспертизы.
- **Автор** — разработчик или создатель элемента экспертизы.
- **Версия** — версия элемента экспертизы, представленная в формате SemVer v2.0. Для неопубликованных элементов экспертизы (черновиков) поле не содержит значений.
- **Теги** — пользовательские теги, присвоенные элементу экспертизы в поле `tags`.
- **Источник данных** — список совместимых с элементом экспертизы источников данных, присвоенных элементу в поле `data_source`.
- **Каталог** — родительский каталог, в котором расположен элемент экспертизы.
- **Статус** — текущий статус элемента экспертизы (включен/выключен).
- **Дата создания** — дата и время создания элемента экспертизы.
- **Создал** — пользователь, инициировавший создание элемента экспертизы.
- **Дата изменения** — дата и время последней модификации элемента экспертизы.
- **Изменил** — пользователь, внесший последние изменения в элемент экспертизы.

Если элемент экспертизы имеет неопубликованные изменения, то в верхней части вкладки **Информация** отображается предупреждение.

Чтобы просмотреть конфигурацию элемента экспертизы:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части вкладки **Информация**. Система отобразит окно просмотра текущей конфигурации.

  В верхнем правом углу открывшегося окна отображается текущая версия элемента экспертизы. Справа от текущей версии расположена кнопка **История**, по которой открывается выпадающий список версий элемента экспертизы. Вы можете просмотреть конфигурацию конкретной версии, выбрав ее из списка.

- Для черновика: нажмите на кнопку **Изменить** в нижней части вкладки **Информация**. Система отобразит окно редактирования текущей конфигурации.

  Окно конфигурации черновика открывается только в режиме редактирования.

Вкладка Связанные элементы

Вкладка **Связанные элементы** отображает список названий элементов экспертизы, к которым обращается текущий элемент или которые обращаются к текущему элементу. Состав списка зависит от типа элемента:

- Для правил нормализации и корреляции: схемы активных списков и таблицы обогащения, к которым обращается правило.
- Для схем активных списков и таблиц обогащения: правила нормализации и корреляции, в которых используется активный список или таблица.
- Для глобальных функций:
  - схемы активных списков и таблицы обогащения, к которым обращается функция;
  - правила нормализации и корреляции, в которых используется функция.

При нажатии на название связанного элемента откроется окно просмотра его конфигурации.

Вкладка Конвейеры

Вкладка **Конвейеры** отображает список конвейеров, на которых используется выбранное правило нормализации, корреляции или агрегации, с возможностью поиска.

Карточка каждого конвейера содержит следующие параметры:

- Название конвейера.
- Идентификатор конвейера.
- Статус конфигурации конвейера с цветовым индикатором:
  - зеленый — конфигурация установлена;
  - серый — конфигурация не установлена;
  - красный — ошибка установки.

- Идентификатор коллектора, в котором расположен конвейер.
- Название коллектора.

При нажатии на кнопку **Конфигурация конвейера** откроется диаграмма конфигурации конвейера, в элементах которого содержится выбранное правило.

Чтобы выйти из диаграммы конфигурации и вернуться на вкладку **Конвейеры** в карточке элемента экспертизы, нажмите на кнопку **Назад** в левой верхней части экрана.

Включение элемента экспертизы

Чтобы опубликованный элемент экспертизы стал доступен на конвейерах событий, его необходимо включить.

Для черновиков возможность включения недоступна.

Чтобы включить элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку элемента экспертизы в списке. Система отобразит в правой части экрана карточку с подробной информацией об элементе.
3. Убедитесь, что в карточке элемента экспертизы открыта вкладка **Информация**.
4. Переведите переключатель состояния элемента экспертизы в верхней части карточки в активное положение. Отобразится окно подтверждения включения элемента экспертизы.
5. Нажмите на кнопку **Включить**. Система включит элемент экспертизы и отобразит соответствующее уведомление. Элемент станет доступным на конвейерах событий.

Если при включении элемента экспертизы возникли ошибки, система отобразит сообщение об этом. Для просмотра списка ошибок нажмите на кнопку **Показать**.

Выключение элемента экспертизы

Чтобы выключить опубликованный элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку элемента экспертизы в списке. Система отобразит в правой части экрана карточку с подробной информацией об элементе.
3. Убедитесь, что в карточке элемента экспертизы открыта вкладка **Информация**.
4. Переведите переключатель в верхней части элемента экспертизы в неактивное положение. Отобразится окно подтверждения выключения элемента экспертизы.
5. Нажмите на кнопку **Выключить**. Система обновит статус элемента экспертизы и отобразит уведомление о выключении. Элемент экспертизы перестанет быть доступным при добавлении элементов конвейера.

Перенос элемента экспертизы

Элементы экспертизы можно переносить между каталогами.

Перенос доступен только для элементов экспертизы с типом создания **Пользовательский**.

Чтобы перенести элемент или группу элементов из списка:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите перенести, в левом столбце таблицы.
3. Нажмите на кнопку **Переместить в каталог** ( ) на панели инструментов. Отобразится окно перемещения каталога.
4. В отобразившемся окне раскройте дерево каталогов и выберите каталог для перемещения.
5. Нажмите на кнопку **Переместить**. Система переместит элемент экспертизы в выбранный каталог и отобразит соответствующее уведомление. Элемент отобразится в выбранном каталоге.

Создание копии элемента экспертизы

Чтобы создать копию элемента экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку элемента экспертизы в списке. Система отобразит в правой части экрана карточку с подробной информацией об элементе.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Создать копию**. Отобразится окно настроек копии элемента экспертизы.

В окне отображаются последние сохраненные для элемента экспертизы настройки конфигурации вне зависимости от того, были ли они опубликованы.

4. Внесите требуемые изменения в конфигурацию копии элемента экспертизы.
5. Нажмите на кнопку **Опубликовать версию**. Система создаст копию элемента и отобразит соответствующее уведомление. Новый элемент экспертизы появится в списке.

   Вы можете создать копию элемента экспертизы в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Система сохранит конфигурацию для дальнейшего редактирования и отобразит соответствующее уведомление. В верхней части карточки элемента будет отображаться предупреждение о наличии неопубликованных изменений.

При создании копии элемента экспертизы требуется изменить его идентификатор в поле `id`.

Копирование кода элемента экспертизы

Чтобы скопировать код элемента экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку элемента экспертизы в списке. Система отобразит в правой части экрана карточку с подробной информацией об элементе.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Скопировать код**. Система скопирует код элемента в буфер обмена и отобразит соответствующее уведомление.

Удаление элемента экспертизы

- Удаление доступно только для элементов экспертизы с типом создания **Пользовательский**.
- Если элемент экспертизы используется в других сущностях системы, его удаление невозможно.

Чтобы удалить элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку элемента экспертизы в списке. Система отобразит в правой части экрана карточку с подробной информацией об элементе.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления элемента экспертизы.
4. Нажмите на кнопку **Удалить**. Система удалит элемент и отобразит соответствующее уведомление. Удаленный элемент будет исключен из списка элементов экспертизы.

Вы также можете удалить группу элементов экспертизы. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные элементы и отобразит соответствующее уведомление. Удаленные элементы будет исключены из списка элементов экспертизы.

Экспорт элементов экспертизы

Экспорт доступен только для элементов экспертизы с типом создания **Пользовательский**.

Чтобы экспортировать элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите экспортировать, в левом столбце таблицы.
3. Нажмите на кнопку **Экспортировать** на панели инструментов. Система запросит подтверждение выполнения экспорта.
4. Нажмите на кнопку **Экспортировать**. Система загрузит элемент экспертизы на устройство пользователя в виде файла с расширением .roc и отобразит соответствующее уведомление.

Неопубликованные изменения и черновики в экспорте не участвуют.

Конфигурацию таблиц обогащения также можно экспортировать из объектного хранилища NATS Object Store.

Импорт элементов экспертизы

Для загрузки элементов экспертизы в систему применяется функционал импорта, поддерживающий загрузку файлов с расширением .roc. Организация каталогов внутри ROC-файла определяет иерархию элементов экспертизы в системе после импорта. Файлы, содержащие элементы экспертизы (такие как правила и списки), должны быть в формате с расширением .roc.

- При импорте архива система автоматически создает каталоги и элементы экспертизы в соответствии со структурой каталогов в архиве.
- Если в системе уже существуют элементы с такими же названиями, как в импортируемом архиве, их содержание обновляется.
- Если в системе уже существуют элементы экспертизы с такими же ID, как в импортируемом архиве, они будут заменены на импортируемые.
- Пути к файлам в системе обновляются согласно новой конфигурации, определенной в архиве.

Чтобы импортировать элемент экспертизы:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Импортировать** на панели инструментов. Система отобразит окно импорта элементов экспертизы.
3. Нажмите **Выбрать файл**. Откроется окно проводника операционной системы.
4. В окне выберите файл для импорта и нажмите на кнопку **Открыть**. Файл отобразится в окне импорта элементов экспертизы.
5. Нажмите на кнопку **Импортировать**. Элемент экспертизы будет загружен в систему. Система отобразит окно с итогами импорта.

В случае возникновения ошибок в ходе импорта, система отобразит список ошибок с указанием ID,

типа и описания ошибки.

Правила корреляции

В данном разделе представлен обзор структуры, принципов работы, а также процесса создания и расширения конфигураций правил корреляции.

О правилах корреляции

Правила корреляции в системе являются основным компонентом коррелятора. Они отвечают за обнаружение взаимосвязанных событий, которые соответствуют заданным критериям и паттернам и также могут инициировать действия после обнаружения, например, создание корреляционных событий.

Правило корреляции представляет собой структурированный объект данных в формате RObject, который содержит конфигурацию для автоматизированного распознания определенных последовательностей обрабатываемых событий ИБ. Оно определяет параметры для фильтрации, группировки, и сопоставления событий, а также содержит логику для пост-обработки корреляционных событий, генерируемых правилом.

В системе поддерживаются два типа корреляционных правил: императивные и декларативные.

Императивные правила корреляции, основанные на алгоритмическом подходе, определяют точную последовательность операций для обработки и анализа данных. Эти правила детально регламентируют процесс, который необходимо выполнить для достижения конкретного результата.

Коллектор, настроенный для работы в режиме распределенной корреляции, не поддерживает работу императивных правил корреляции.

Поддержка императивных правил корреляции будет прекращена в ближайших версиях системы. При разработке новых правил корреляции рекомендуется использовать декларативный подход.

Декларативные правила корреляции ориентированы на описание желаемого результата без указания конкретных шагов для его достижения. Эти правила фокусируются на определении условий, при которых должен произойти определенный вывод, позволяя системе самостоятельно определять путь достижения результата. Декларативные правила обеспечивают более высокий уровень абстракции и упрощают процесс определения корреляционных связей.

Правила корреляции создаются с использованием языка преобразований и корреляции VRL, который позволяет формулировать сложные логические условия и трансформации данных для анализа и корреляции событий.

Работа с правилом корреляции

Доступные операции над правилом корреляции:

- Создание правила корреляции
- Изменение правила корреляции

- Удаление правила корреляции

Создание правила корреляции

Создать правило корреляции можно через его настройку в конструкторе или через задание его RObject-конфигурации.

Создание через конструктор доступно только для декларативных правил корреляции. Создание императивных правил выполняется только через задание их RObject-конфигурации.

Создание правила корреляции через конструктор

Конструктор правил корреляции позволяет визуально построить правило корреляции в интерактивном режиме без настройки его конфигурации.

Чтобы добавить правило корреляции:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило корреляции**. Система отобразит окно выбора способа создания правила.
3. Выберите способ создания правила через конструктор и нажмите на кнопку **Создать**. Система отобразит окно конструктора правил корреляции.
4. Заполните поля правила корреляции в конструкторе, чтобы определить логику работы правила.
5. Нажмите на кнопку **Опубликовать**. Система создаст правило корреляции и отобразит соответствующее уведомление. Новое правило отобразится в таблице раздела **Экспертиза**.

Чтобы правило корреляции стало доступно на конвейерах событий, его необходимо включить.

Вы можете создать черновик правила корреляции без публикации с помощью кнопки **Сохранить как черновик**. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки правила будет отображаться предупреждение о наличии неопубликованных изменений.

Создание RObject-конфигурации правила корреляции

Чтобы добавить правило корреляции:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило корреляции**. Система отобразит окно выбора способа создания правила.
3. Выберите способ создания правила через код и нажмите на кнопку **Создать**. Система отобразит окно задания RObject-конфигурации правила корреляции.

При создании элемента экспертизы поля его структуры заполняются

значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

4. Заполните поля правила корреляции, чтобы определить логику его работы.
5. Нажмите на кнопку **Опубликовать версию**. Система создаст правило корреляции и отобразит соответствующее уведомление. Новое правило появится в таблице раздела **Экспертиза**.

Чтобы правило корреляции стало доступно на конвейерах событий, его необходимо включить.

Вы можете создать черновик правила корреляции без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик правила и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Изменение правила корреляции

Изменить правило корреляции можно с помощью изменения его настроек в конструкторе или через редактирование его RObject-конфигурации.

Изменение через конструктор доступно только для декларативных правил корреляции. Изменение императивных правил выполняется только через редактирование их RObject-конфигурации.

Изменение правила корреляции в конструкторе

Конструктор правил корреляции позволяет визуально построить правило корреляции в интерактивном режиме без настройки его конфигурации.

Чтобы изменить правило корреляции в конструкторе:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку правила корреляции в списке. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить в конструкторе**. Система отобразит окно конструктора правил корреляции.

Вы также можете открыть окно конструктора с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек правила. Нажмите на кнопку **Изменить в конструкторе** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки. Система отобразит окно настройки конфигурации правила. Перейдите в режим конструктора с помощью переключателя в правом верхнем углу

окна.

4. Внесите требуемые изменения в настройки правила корреляции в конструкторе.
5. Нажмите на кнопку **Обновить версию**. Система опубликует новую версию правила корреляции с измененной конфигурацией и отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию правила корреляции в виде черновика без публикации с помощью кнопки **Сохранить как черновик**. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки правила будет отображаться предупреждение о наличии неопубликованных изменений.

Изменение RObject-конфигурации правила корреляции

Чтобы изменить RObject-конфигурацию правила корреляции:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку правила корреляции в списке. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить код**. Отобразится окно настроек RObject-конфигурации правила.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек правила. Нажмите на кнопку **Изменить в коде** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки. Система отобразит окно настройки конфигурации правила.

4. Внесите требуемые изменения в конфигурацию правила корреляции.
5. Нажмите на кнопку **Обновить версию**. Система опубликует новую версию правила корреляции с измененной конфигурацией и отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию правила корреляции в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки правила будет отображаться предупреждение о наличии неопубликованных изменений.

Удаление правила корреляции

- Удаление доступно только для элементов экспертизы с типом создания **Пользовательский**.
- Если элемент экспертизы используется в других сущностях системы, его удаление невозможно.

Чтобы удалить правило корреляции:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку правила корреляции в списке. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления правила.
4. Нажмите на кнопку **Удалить**. Система удалит правило корреляции и отобразит соответствующее уведомление. Удаленное правило будет исключено из списка элементов экспертизы.

Вы также можете удалить группу элементов экспертизы. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные элементы и отобразит соответствующее уведомление. Удаленные элементы будет исключены из списка элементов экспертизы.

Построение правила корреляции в конструкторе

Конструктор правил корреляции позволяет визуально построить правило корреляции в интерактивном режиме без необходимости ручной настройки его RObject-конфигурации. Конструктор автоматически создает правило нужной конфигурации на основе заданных настроек.

Построение через конструктор доступно только для декларативных правил корреляции. Создание или изменение императивных правил выполняется только через настройку их RObject-конфигурации.

Построение правила корреляции с помощью конструктора состоит из следующих этапов:

1. Заполнение общей информации о правиле
2. Настройка фильтрации событий
3. Настройка объединения событий
4. Настройка корреляционных событий

5. Настройка тестов
6. Сохранение правила

Заполнение общей информации о правиле

На вкладке **Общая информация** в конструкторе заполняются поля общей информации о правиле корреляции.

Чтобы задать общую информацию о правиле, заполните следующие поля:

1. **ID** — введите уникальный идентификатор правила корреляции. Указанное значение будет сохранено в поле `id` в RObject-конфигурации правила.
2. **Версия** — введите версию правила корреляции в формате Semantic Versioning. Указанное значение будет сохранено в поле `version` в RObject-конфигурации правила.
3. **Название** — введите название правила корреляции. Указанное значение будет сохранено в поле `name` в RObject-конфигурации правила.
4. **Описание** (опционально) — введите описание правила корреляции, содержащее дополнительную информацию о его назначении и функциональности. Указанное значение будет сохранено в поле `description` в RObject-конфигурации правила.
5. **Автор** (опционально) — введите имя и контактные данные автора правила корреляции. Указанное значение будет сохранено в поле `author` в RObject-конфигурации правила.
6. **Статус** (опционально) — введите статус разработки варианта правила. Указанное значение будет сохранено в поле `status` в RObject-конфигурации правила.
7. **Дата** — при необходимости выберите другую дату создания или изменения правила. По умолчанию установлена текущая дата. Указанное значение будет сохранено в поле `date` в RObject-конфигурации правила.
8. **Уровень угрозы** — выберите уровень угрозы корреляционного события, испускаемого правилом корреляции. Доступные значения:
   - Критический (`critical`);
   - Высокий (`high`);
   - Средний (`medium`);
   - Низкий (`low`).

   Выбранное значение будет сохранено в поле `severity` в RObject-конфигурации правила.
9. **Источники данных** (опционально) — введите список источников данных, с которыми совместимо правило корреляции. Данный список используется для классификации и быстрого поиска правил в системе. Указанное значение будет сохранено в поле `data_source` в RObject-конфигурации правила.
10. **Референсы** (опционально) — введите ссылки на подробный материал. Указанное значение будет сохранено в поле `reference` в RObject-конфигурации правила.
11. **Известные ложные срабатывания** (опционально) — введите возможные ложноположительные срабатывания. Указанное значение будет сохранено в поле `known_false_positives` в RObject-конфигурации правила.

12. **Теги** (опционально) — введите список тегов для правила корреляции. Указанное значение будет сохранено в поле `tags` в RObject-конфигурации правила.
13. **Максимальное количество корреляционных окон** (опционально) — введите максимально допустимое количество корреляционных окон, которое можно запустить одновременно. При превышении система отключает правило корреляции. Указанное значение будет сохранено в поле `max_correlation_windows` в RObject-конфигурации правила.
14. **Ограничения генерации корреляционных событий** (опционально) — установите переключатель в активное положение, чтобы настроить параметры для ограничения генерации корреляционных событий. При превышении ограничений система отключает правило корреляции. Указанные значения будут сохранены в полях группы `rate_limit` в RObject-конфигурации правила.

    Доступны следующие параметры:

    - **Количество событий** — максимально допустимое количество корреляционных событий, генерируемых за указанный период времени. Соответствует полю `max_rate` группы `rate_limit` в RObject-конфигурации правила.
    - **Интервал для генерации событий** — промежуток времени в секундах, в течение которого допустима генерация не более указанного количества корреляционных событий. Соответствует полю `duration_secs` группы `rate_limit` в RObject-конфигурации правила.
    - **Время ожидания восстановления работы правила** — промежуток времени в секундах, в течение которого ожидается восстановление нормального режима работы правила. Если режим работы правила не восстановится, система отключит его. Соответствует полю `grace_period_secs` группы `rate_limit` в RObject-конфигурации правила.

Пусть установлены следующие ограничения генерации корреляционных событий:

- **Количество событий** — 10.
- **Интервал для генерации событий** — 60 сек.
- **Время ожидания восстановления работы правила** — 60 сек.

Предположим, правило сгенерировало более 10 событий за 60 секунд. Если за последующие 60 секунд также будет сгенерировано более 10 событий, система отключит правило корреляции.

Настройка фильтрации событий

На вкладке **Фильтры событий** в конструкторе настраивается фильтр, определяющий условия, при которых правило корреляции будет применяться к событиям.

Чтобы настроить фильтрацию событий:

1. Добавьте псевдонимы событий.
2. Задайте критерии фильтрации событий.
3. При необходимости настройте группировку событий для псевдонимов.

Настройка псевдонимов событий

Псевдонимы потоков событий используются для их идентификации в рамках правила корреляции, а также для их разделения и фильтрации. В RObject-конфигурации правила корреляции настройки псевдонимов задаются в поле `aliases`.

Чтобы добавить псевдоним событий в конструкторе:

1. Нажмите на кнопку **Добавить событие** (+). Отобразятся настройки псевдонима событий.
2. При необходимости измените название псевдонима. По умолчанию псевдонимы создаются с названием "event_<номер псевдонима>".
3. Задайте критерии фильтрации событий.
4. При необходимости настройте группировку событий для псевдонима.

Чтобы удалить псевдоним событий, нажмите на кнопку 🗑 в строке его заголовка.

На вкладке **Фильтры событий** должен быть задан хотя бы один псевдоним. Удаление последнего псевдонима недоступно.

Настройка фильтрации

Фильтрация может задаваться как для конкретных псевдонимов событий, так и для всего потока событий:

- Фильтрация псевдонима настраивается с помощью условий, задаваемых в области настройки псевдонима. Для каждого псевдонима должно быть задано хотя бы одно условие фильтрации.
- Фильтрация всего потока событий выполняется с помощью общего фильтра. Чтобы включить общий фильтр, необходимо перевести переключатель **Общий фильтр** в активное положение. В RObject-конфигурации правила корреляции общий фильтр задается в поле `filter`.

Чтобы добавить условие фильтрации для псевдонима событий или общего фильтра:

1. Добавьте поля настроек условия. Способ добавления настроек отличается в зависимости от того, является ли добавляемое условие первым в псевдониме или общем фильтре:
   - Если вы добавляете первое условие, нажмите на кнопку **Добавить условие** (+) и выберите тип условия. Отобразятся поля настроек выбранного типа условия.
   - Если псевдоним или общий фильтр уже содержит условия, новые условия добавляются с использованием логической связи с существующими условиями. Чтобы добавить условие с логической связью, нажмите на кнопку необходимой логической связи в строке нужного условия и выберите тип условия.

     Доступные кнопки логической связи:

     - **И** (⊕) — позволяет добавить новое условие к выбранному условию и связать их через логическое "И".

       Если выбранное условие связано с другими условиями на том же уровне через логическое "ИЛИ", то новое условие создаст подгруппу вместе с выбранным условием. Условия внутри созданной подгруппы будут связаны через логическое "И". Сама группа

будет связана с условиями на своем уровне через логическое "ИЛИ".

Если выбранное условие связано с другими условиями на том же уровне через логическое "И", то новое условие будет добавлено на тот же уровень без создания подгруппы.

- **ИЛИ** (⊕) — позволяет добавить новое условие к выбранному условию и связать их через логическое "ИЛИ".

  Если выбранное условие связано с другими условиями на том же уровне через логическое "И", то новое условие создаст подгруппу вместе с выбранным условием. Условия внутри созданной подгруппы будут связаны через логическое "ИЛИ". Сама группа будет связана с условиями на своем уровне через логическое "И".

  Если выбранное условие связано с другими условиями на том же уровне через логическое "ИЛИ", то новое условие будет добавлено на тот же уровень без создания подгруппы.

Нельзя добавить более трех уровней условий. Добавление новых условий к группе на третьем уровне выполняется через тип связи, уже установленный в этой группе. Например, если в группе на третьем уровне условия связаны между собой через логическое "И", то новое условие в этой группе может быть добавлено только со связью "И".

2. Задайте настройки условия. Настройки отличаются в зависимости от типа условия.

При сохранении правила корреляции через конструктор все заданные условия преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Чтобы удалить условие фильтрации, нажмите на кнопку 🗑 в его строке.

Для каждого псевдонима событий должно быть задано хотя бы одно условие фильтрации.

Настройка условий разных типов

Доступные типы условий фильтрации:

- Сравнение полей
- Запрос в активный список
- Запрос в таблицу обогащения
- Глобальная функция
- Блок кода

Настройка условия Сравнение полей

Условие типа **Сравнение полей** позволяет сравнивать значение поля в событии со значением другого поля или с каким-либо заданным значением.

При сохранении правила корреляции через конструктор все заданные условия преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Чтобы настроить условие типа **Сравнение полей**:

1. Выберите из выпадающего списка поле события.

На выбор доступны только поля универсальной модели события, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

2. При необходимости выберите из выпадающего списка тип данных выбранного поля: DateTime, String, LCString, UInt, Float, UUID, Bool, IPv4, IPv6, MAC, KeyValue, Array, Enum.

Изменение типа данных для полей универсальной модели события позволяет переопределить их системный тип данных в рамках заданного условия.

3. Выберите оператор сравнения из выпадающего списка:
   - `=` — равно.
   - `!=` — не равно.
   - `In` (только для строки) — поиск подстрок. Строка, в которой будет осуществляться поиск, должна быть указана в формате JSON.
   - `Like` (только для строки) — поиск строк по шаблону.

Шаблон может включать обычные символы и следующие метасимволы:

- `%` — обозначает любое количество произвольных символов или отсутствие символов.
- `_` — обозначает один произвольный символ.
- `\` — используется для экранирования символов `%`, `_` и `\`.

   - `=null` — не имеет значений (пустое поле).
   - `!=null` — имеет значение (непустое поле).

4. Выберите тип значения из выпадающего списка:
   - **Значение** — потребуется указать значение в отобразившемся поле. Способ задания значения зависит от типа данных поля.

     Значение исходного поля будет сравниваться с указанным значением.

Если выбран оператор `In`, экранировать специальные символы не требуется, за исключением двойных кавычек внутри строковых значений.

Например, для строки `["a", "b"]` дополнительное экранирование не требуется, а для строки `["a", "\"b"\"]` требуется экранировать кавычки, в которые заключен символ `b`.

- **Значение из поля** — потребуется выбрать поле события из выпадающего списка.

На выбор доступны только поля универсальной модели события, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

Значение исходного поля будет сравниваться со значением указанного поля.

Выбор типа значения недоступен при выборе оператора `=null` или `!=null`.

Настройка условия Запрос в активный список

Условие типа **Запрос в активный список** позволяет дополнить поток событий в псевдониме записями активного списка, которые могут использоваться для обогащения корреляционных событий.

При сохранении правила корреляции через конструктор все заданные условия преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Чтобы настроить условие типа **Запрос в активный список**:

1. Выберите нужный активный список из выпадающего списка. В области настроек условия отобразятся все поля ключей выбранного активного списка.
2. Выберите условие из выпадающего списка:
   - **Есть запись, для которой выполняются все условия** — условие истинно, если в активном списке существует запись, удовлетворяющая всем заданным условиям.
   - **Нет записей, для которых выполняются все условия** — условие истинно, если в активном списке нет записей, удовлетворяющих всем заданным условиям.
3. Для каждого поля ключа активного списка выберите тип значения из выпадающего списка:
   - **Значение** — потребуется указать значение в отобразившемся поле. Способ задания значения зависит от типа данных поля.

     Значение поля ключа будет сравниваться с указанным значением.

   - **Значение из поля** — потребуется выбрать поле события из выпадающего списка.

На выбор доступны только поля универсальной модели события, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

Значение поля ключа будет сравниваться со значением указанного поля события.

Настройка условия Запрос в таблицу обогащения

Условие типа **Запрос в таблицу обогащения** позволяет дополнить поток событий в псевдониме записями таблицы обогащения, которые могут использоваться для обогащения корреляционных событий.

При сохранении правила корреляции через конструктор все заданные условия преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Чтобы настроить условие типа **Запрос в таблицу обогащения**:

1. Выберите нужную таблицу обогащения из выпадающего списка.
2. Выберите условие из выпадающего списка:
   - **Есть запись, для которой выполняются все условия** — условие истинно, если в таблице обогащения существует запись, удовлетворяющая всем заданным условиям.
   - **Нет записей, для которых выполняются все условия** — условие истинно, если в таблице обогащения нет записей, удовлетворяющих всем заданным условиям.
3. Добавьте условия для таблицы обогащения.

   Чтобы добавить условие:

   a. Нажмите на кнопку **Добавить поле** (＋). Добавится строка настройки условия.

   По умолчанию при выборе таблицы обогащения уже отображается одна строка настройки условия.

   b. Выберите из выпадающего списка поле таблицы обогащения, для которого будет задаваться условие.

   c. Выберите тип значения поля из выпадающего списка:

   - **Значение** — потребуется указать значение в отобразившемся поле. Способ задания значения зависит от типа данных поля.

     Значение поля таблицы будет сравниваться с указанным значением.

   - **Значение из поля** — потребуется выбрать поле события из выпадающего списка.

   На выбор доступны только поля универсальной модели события, а также поля, ранее добавленные вручную.

   Чтобы вручную добавить поле в выпадающий список:

   i. Введите название поля.

   ii. Выберите опцию **Создать**.

Значение поля таблицы будет сравниваться со значением указанного поля события.

Чтобы удалить условие, нажмите на кнопку в его строке.

Для таблицы обогащения должно быть задано хотя бы одно условие. Удаление последнего условия недоступно.

Настройка условия Глобальная функция

Условие типа **Глобальная функция** позволяет использовать глобальные функции для фильтрации событий.

При сохранении правила корреляции через конструктор все заданные условия преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Чтобы настроить условие типа **Глобальная функция**:

1. Выберите нужную глобальную функцию из выпадающего списка. В области настроек условия отобразятся все параметры выбранной глобальной функции.
2. Для каждого параметра глобальной функции выберите тип значения из выпадающего списка:
   - **Значение** — потребуется указать значение в отобразившемся поле. Способ задания значения зависит от типа данных параметра.
   - **Значение из поля** — потребуется выбрать поле события из выпадающего списка.

На выбор доступны только поля универсальной модели события, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

Настройка условия Блок кода

Условие типа **Блок кода** позволяет задать условие в виде кода на языке VRL.

При сохранении правила корреляции через конструктор все заданные условия преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Чтобы настроить условие типа **Блок кода**, задайте VRL-код условия.

В VRL-программе для доступа к данным события используется символ точки (`.`), а к метаданным события — символ процента (`%`). В корреляционных событиях метаданные включают данные исходных событий.

Настройка группировки событий

Группировка позволяет объединять события в группы. В RObject-конфигурации правила корреляции группировка задается в поле `group`.

Чтобы добавить группировку:

1. Нажмите на кнопку **Добавить группировку** (+). Отобразятся настройки группировки.
2. Задайте количество событий в группе вручную или с помощью кнопок — и +. Допустимый диапазон: 2—100.
3. Выберите нужные поля события из выпадающего списка.

На выбор доступны только поля универсальной модели события, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

4. Если необходимо учитывать только уникальные события в рамках указанных полей группировки, установите флажок **Уникальность событий в рамках группировки**.

Вы можете добавить не более двух группировок. При этом одна группировка должна быть с уникальностью событий, другая — без уникальности событий.

Чтобы удалить группировку, нажмите на кнопку в строке ее заголовка.

Настройка объединения событий

На вкладке **Объединение** задаются правила объединения двух или более потоков событий. В RObject-конфигурации правила корреляции настройки объединения задаются в поле `select`.

Чтобы настроить объединения:

1. Укажите время действия (TTL) корреляционного окна в секундах. В RObject-конфигурации правила корреляции этой настройке соответствует поле `ttl`.
2. Если на вкладке **Фильтры событий** добавлено более одного псевдонима событий:
   a. Выберите из выпадающего списка один из псевдонимов.
   b. Добавьте объединение для выбранного псевдонима событий.
   c. Если на вкладке **Фильтры событий** добавлено более двух псевдонимов событий, добавьте объединения для оставшихся псевдонимов.

Настройка объединений

Добавление объединений недоступно, если на вкладке **Фильтры событий** добавлен только один псевдоним событий.

В RObject-конфигурации правила корреляции настройка объединения выполняется в поле `join`.

Чтобы добавить объединение:

1. Нажмите на кнопку **Добавить объединение** (+). Отобразятся настройки объединения.

2. Выберите тип объединения для определения условий формирования корреляционного события:
   - **Получение события** — за указанный таймаут должно быть получено соответствующее событие. В RObject-конфигурации правила корреляции этой настройке соответствует поле `absent` со значением `false`.
   - **Отсутствие события** — за указанный таймаут не должно быть получено соответствующее событие. В RObject-конфигурации правила корреляции этой настройке соответствует поле `absent` со значением `true`.
3. Введите время ожидания (таймаут) получения соответствующего события в секундах. В RObject-конфигурации правила корреляции этой настройке соответствует поле `wait_for`.
4. Выберите из выпадающего списка один из псевдонимов событий, добавленных на вкладке **Фильтры событий**.

На выбор доступны только те псевдонимы, которые ранее не использовались на вкладке **Объединение**.

5. Добавьте условия сопоставления полей.

Чтобы удалить объединение, нажмите на кнопку в строке заголовка объединения.

Настройка сопоставления полей

Сопоставление полей позволяет сравнить определенные поля в событии исходного псевдонима и в событии другого псевдонима. В зависимости от итогов этого сравнения за указанный таймаут и от заданных условий формирования корреляционного события, по итогам выполнения корреляции будет создаваться корреляционное событие.

Чтобы добавить условие сопоставления полей в объединении:

1. Нажмите на кнопку **Добавить условие** (+). Отобразится строка для задания условия сопоставления полей.
2. Выберите из выпадающего списка псевдоним, поля событий которого требуется сопоставлять с полями событий исходного псевдонима.

- На выбор доступны только те псевдонимы, которые ранее уже использовались на вкладке **Объединение** для других объединений.
- Для первого объединения автоматически устанавливается псевдоним, выбранный в верхней части вкладки **Объединение**.

3. Выберите из выпадающего списка поле события.

На выбор доступны только поля универсальной модели события, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

Указанные поля в событиях выбранных псевдонимов будут участвовать в сопоставлении полей.

Чтобы удалить условие сопоставления полей в объединении, нажмите на кнопку 🗑 в строке условия.

В объединении должно быть хотя бы одно условие сопоставления полей. Удаление последнего условия в объединении недоступно.

Настройка корреляционных событий

На вкладке **Настройка события корреляции** определяются действия, которые должны выполняться при успешной корреляции событий, настраиваются правила формирования корреляционных событий, а также задаются прочие дополнительные условия корреляции. В RObject-конфигурации правила корреляции эти настройки задаются в поле `on_correlate`.

Чтобы настроить поведение корреляционных событий:

1. При необходимости введите время троттлинга в секундах. Время троттлинга определяет временной интервал, который указывает, как часто могут генерироваться корреляционные события для каждой группы, заданной на вкладке **Объединение**. Если групп нет, время троттлинга указывается глобально на все правило. Указанное значение будет сохранено в поле `throttle_time_sec` в RObject-конфигурации правила.
2. Сформируйте список полей корреляционного события.
3. При необходимости введите правила формирования текста сообщения в корреляционном правиле на языке VRL.
4. При необходимости введите дополнительные условия корреляции на языке VRL.

При сохранении правила корреляции через конструктор все заданные настройки корреляции преобразуются из блока `!vrl` в блок `!expression` в RObject-конфигурации правила.

Формирование полей корреляционного события

Чтобы добавить поле в корреляционное событие:

1. Нажмите на кнопку **Добавить поле** (+). Отобразится строка настройки нового поля.
2. Выберите из выпадающего списка поле события.

На выбор доступны только поля универсальной модели события, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

3. Выберите из выпадающего списка тип источника, из которого поле будет брать значение.
4. Задайте настройки источника. Настройки отличаются в зависимости от типа источника.

Чтобы удалить поле, нажмите на кнопку в строке его настроек.

Настройка источников разных типов

Доступные типы источников:

- Значение
- Значение из поля
- Запрос в активный список
- Запрос в таблицу обогащения

Настройка источника типа Значение

Источник типа **Значение** позволяет задать определенное значение, которое будет присвоено полю корреляционного события.

Чтобы настроить источник типа **Значение**, укажите значение в отобразившемся поле. Способ задания значения зависит от типа данных поля.

Настройка источника типа Значение из поля

Источник типа **Значение из поля** позволяет задать поле события, значение которого будет использоваться для значения поля в корреляционном событии.

Чтобы настроить источник типа **Значение из поля**:

1. Выберите из выпадающего списка псевдоним события. На выбор доступны псевдонимы, заданные на вкладке **Фильтры событий**.
2. Выберите из выпадающего списка поле события.

На выбор доступны только поля универсальной модели события, а также поля, ранее добавленные вручную.

Чтобы вручную добавить поле в выпадающий список:

i. Введите название поля.

ii. Выберите опцию **Создать**.

Настройка источника типа Запрос в активный список

Источник типа **Запрос в активный список** позволяет найти запись в активном списке и указать поле записи, значение которого будет использоваться для значения поля в корреляционном событии.

Чтобы настроить источник типа **Запрос в активный список**:

1. Выберите нужный активный список из выпадающего списка. В области настроек источника отобразятся все поля ключей выбранного активного списка.
2. Для каждого поля ключа активного списка выберите тип значения из выпадающего списка:
   - **Значение** — потребуется указать значение в отобразившемся поле. Способ задания значения

зависит от типа данных поля.

- **Значение из поля** — потребуется выбрать псевдоним события. На выбор доступны псевдонимы, заданные на вкладке **Фильтры событий**.

  Затем необходимо выбрать поле события из выпадающего списка.

  На выбор доступны только поля универсальной модели события, а также поля, ранее добавленные вручную.

  Чтобы вручную добавить поле в выпадающий список:

  i. Введите название поля.

  ii. Выберите опцию **Создать**.

  Значение выбранного поля в выбранном событии будет использоваться в качестве значения поля активного списка.

3. Выберите из выпадающего списка поле активного списка, значение которого будет использоваться для значения поля в корреляционном событии.

Настройка источника типа Запрос в таблицу обогащения

Источник типа **Запрос в таблицу обогащения** позволяет найти запись в таблице обогащения и указать поле записи, значение которого будет использоваться для значения поля в корреляционном событии.

1. Выберите нужную таблицу обогащения из выпадающего списка.
2. Добавьте поля таблицы обогащения, по которым будет осуществляться поиск записи в таблице.

   Чтобы добавить поле:

   a. Нажмите на кнопку **Добавить поле** (＋). Добавится строка настройки поля.

   b. Выберите из выпадающего списка поле таблицы обогащения.

   c. Выберите тип значения поля из выпадающего списка:

   - **Значение** — потребуется указать значение в отобразившемся поле. Способ задания значения зависит от типа данных поля.
   - **Значение из поля** — потребуется выбрать псевдоним события. На выбор доступны псевдонимы, заданные на вкладке **Фильтры событий**.

     Затем необходимо выбрать поле события из выпадающего списка.

     На выбор доступны только поля универсальной модели события, а также поля, ранее добавленные вручную.

     Чтобы вручную добавить поле в выпадающий список:

     i. Введите название поля.

     ii. Выберите опцию **Создать**.

Значение выбранного поля в выбранном событии будет использоваться в качестве значения поля таблицы обогащения.

Чтобы удалить поле, нажмите на кнопку в его строке.

Для таблицы обогащения должно быть задано хотя бы одно поле. Удаление последнего поля недоступно.

3. Выберите из выпадающего списка поле таблицы обогащения, значение которого будет использоваться для значения поля в корреляционном событии.

Настройка тестов

На вкладке **Тесты** опционально задаются тесты для проверки и верификации правила корреляции. Они помогают убедиться, что правило работает корректно и соответствует ожидаемым результатам.

Тесты, заданные на вкладке **Тесты**, заполняют блок `tests` в RObject-конфигурации правила корреляции.

Чтобы добавить тесты:

1. Введите набор тестов на языке VRL.
2. Нажмите на кнопку **Запустить** для запуска выполнения программы.

   Если выполнение программы прошло успешно, обработанное событие отобразится на панели **Результат**.

Сохранение правила

Вы можете сохранить созданное правило корреляции по одной из следующих кнопок в правом нижнем углу конструктора:

- **Опубликовать** — правило будет сохранено и опубликовано. Система создаст правило корреляции и отобразит соответствующее уведомление. Новое правило отобразится в таблице раздела **Экспертиза**. Опубликованные правила можно включить и использовать в конвейерах событий.
- **Сохранить как черновик** — правило будет сохранено в виде черновика без публикации. Система создаст черновик правила корреляции и отобразит соответствующее уведомление. Черновик отобразится в таблице раздела **Экспертиза**. В отличие от опубликованных версий, черновики не могут быть использованы в конвейерах событий.

Императивные правила

В данном разделе представлен обзор структуры, принципа работы, процесса создания и расширения конфигурации императивного правила корреляции. Управление правилами корреляции в системе осуществляется в разделе **Экспертиза**.

Поддержка императивных правил корреляции будет прекращена в ближайших версиях системы. При разработке новых правил корреляции рекомендуется использовать декларативный подход.

Об императивных правилах корреляции

Императивное правило корреляции в системе представляет собой объект данных в формате RObject (.ro), содержащий детально определенные команды и процедуры для анализа и корреляции событий ИБ. Такие правила характеризуются точной спецификацией действий, определенным порядком операций, контрольными структурами и явным управлением состоянием.

Коллектор, настроенный для работы в режиме распределенной корреляции, не поддерживает работу императивных правил корреляции.

Императивное правило корреляции состоит из набора полей, которые можно разделить на два типа:

- Набор общих полей, которые предоставляют основную информацию о правиле и его назначении, задают контекст для корреляции и определяют основные характеристики правила. Этот набор общих полей остается неизменным независимо от типа корреляционного правила.
- Набор дополнительных полей, который изменяется в зависимости от выбранного режима работы окна корреляции.

Императивные правила корреляции поддерживают три режима работы окна корреляции (`window_mode`):

- Параллельный (`parallel`) — это режим работы корреляционного окна, при котором создаваемые правилом окна могут открываться друг за другом и работать одновременно, накладываясь друг на друга во времени. Таким образом, одно событие может присутствовать сразу в нескольких окнах одновременно. Этот метод позволяет системе обнаруживать сложные атаки и аномалии, так как события могут анализироваться одновременно в рамках различных временных контекстов.

Если поле `start_when` не указано, окно будет открываться для каждого события, прошедшего через фильтр. В таком случае событие также будет добавлено в каждое открытое корреляционное окно.

- Без перекрывания (`non-overlapping`) — это режим работы корреляционного окна, при котором правило имеет одновременно не более одного активного окна, таким образом открываемые правилом окна не пересекаются во времени. Окно создается при получении первого события, которое удовлетворяет условиям открытия окна. Последующее окно может быть открыто только после закрытия текущего окна. Таким образом, каждое событие может находиться не более чем в одном окне одновременно. Этот подход предотвращает дублирование анализа событий и обеспечивает предсказуемое распределение ресурсов для обработки данных.
- Без состояния (`stateless`) — это режим работы корреляционного окна, при котором создаваемые правилом окна хранят данные только об одном событии. На каждое событие, удовлетворившее условию, создается индивидуальное корреляционное окно, которое сразу закрывается, генерируя корреляционное событие. Таким образом, каждое событие анализируется независимо от предыдущего, и их контекст не учитывается при анализе. Этот режим обеспечивает простоту реализации и минимальную нагрузку на систему, но ограничивает способность обнаружения сложных атак и аномалий, требующих анализа последовательности событий в контексте периода времени.

Правила корреляции создаются с использованием языка преобразований и корреляции VRL,

который позволяет формулировать сложные логические условия и трансформации данных для анализа и корреляции событий.

Режим работы окна корреляции определяется в поле `window_mode`. Каждое правило может функционировать только в одном выбранном режиме. Если режим не задан, система ожидает декларативный подход к написанию правила.

Структура правила корреляции

Правила корреляции содержат набор обязательных и опциональных полей. Состав полей и логика их работы отличаются в зависимости от заданного режима работы корреляционного окна.

Общие поля

Поля верхнего уровня:

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор правила корреляции.	строка	да
`name`	Название правила, которое обозначает его цель или функцию.	строка	да
`version`	Версия правила корреляции, представленная в формате Semantic Versioning.	строка	да
`author`	Имя и контактные данные автора правила корреляции.	строка	нет
`window_mode`	Режим окна корреляции, определяющий каким образом система будет обрабатывать корреляционные события. Доступные режимы: `parallel`, `non-overlapping`, `stateless`. Если значение не задано, система ожидает декларативный подход к написанию правила.	строка	да
`description`	Описание правила корреляции, где можно указать дополнительную информацию о его назначении и функциональности.	строка	нет
`data_source`	Список источников данных, с которыми совместимо правило корреляции. Используется для классификации и быстрого поиска элементов экспертизы в системе.	массив строк массив объектов	нет
`tags`	Список тегов, присвоенных правилу корреляции.	массив строк	нет
`group_by`	События группируются по этим полям и каждая группа коррелируется независимо. Значения полей группы автоматически попадают в поля корреляционного события.	массив строк	нет

Поле	Описание	Тип данных	Обязательное поле
`type`	Тип правила. Всегда имеет значение `correlation_rule` для правила этого типа.	строка	да
`severity`	Уровень угрозы корреляционного события, испускаемого корреляционным правилом. Доступны значения: `low`, `medium`, `high`, `critical`.	строка	да
`filter`	Блок правил на языке VRL, определяющий условия (фильтр), при которых правило корреляции будет применяться к событиям.	строка	нет
`on_correlation`	Блок правил на языке VRL, определяющий логику обработки корреляционного события после его создания.	строка	нет
`throttle_time_sec`	Указывает интервал в секундах, который определяет, как часто могут генерироваться корреляционные события для каждой группы в поле `group_by`.	целое число	нет
`tests`	Блок тестирования правила корреляции, содержащий массив тестовых событий и условия для проверки срабатывания правила. Обязательные поля: `name`, `events`, `assertion`.	объект	нет

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform> # Платформа или информационная система.
- <source> # Источник событий.
# События (EventId) или путь к журналу событий:
- <EventID_1>
- <EventID_2>
```

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `tests`

Поле	Описание	Тип данных	Обязательное поле
`name`	Название теста. Позволяет идентифицировать конкретный тест в тестовой выдаче.	строка	да
`events`	Поток тестовых событий, поступающих в тестируемый элемент конвейера.	строка	да
`assertion`	Блок правил на языке VRL, который будет вызываться на каждое событие, испускаемое полем `events`. Используя встроенные функции `assert` и `assert_eq`, можно строить проверки на соответствие испускаемого события тестовым условиям.	строка	да

Дополнительные поля режима `parallel`

Поле	Описание	Тип данных	Обязательное поле
`on_event`	Блок правил на языке VRL, определяющий логику обновления состояния окна корреляции при получении событий.	строка	да
`correlate_when`	Блок правил на языке VRL, определяющий условие закрытия корреляционного окна и генерации корреляционного события.	строка	да
`start_when`	Блок правил на языке VRL, определяющий условие открытия корреляционного окна. Если условие не задано, новое окно будет открываться на каждое событие, прошедшее через фильтр.	строка	нет
`expire_after_sec`	Время жизни корреляционного окна в секундах, если не происходит обновление его состояния.	целое число	нет
`windows_limit`	Максимально допустимое количество корреляционных окон, которое можно открыть одновременно.	целое число	нет

Дополнительные поля режима `non-overlapping`

Поле	Описание	Тип данных	Обязательное поле
`on_event`	Блок правил на языке VRL, определяющий логику обновления состояния окна корреляции при получении событий.	строка	да

Поле	Описание	Тип данных	Обязательное поле
`correlate_when`	Блок правил на языке VRL, определяющий условие закрытия корреляционного окна и испускания корреляционного события.	строка	да
`expire_after_sec`	Указывает время автоматического закрытия корреляционного окна (по умолчанию 60 сек).	целое число	нет

Дополнительные поля режима `stateless`

Поле	Описание	Тип данных	Обязательное поле
`on_event`	Блок правил на языке VRL, определяющий логику обновления состояния окна корреляции при получении событий. Если событие не обновляет окно, необходимо вызвать инструкцию `abort`.	строка	нет
`correlate_when`	Блок правил на языке VRL, определяющий условие закрытия корреляционного окна и испускания корреляционного события.	строка	да

Составление правила

Чтобы составить правило корреляции:

1. Заполните общие поля метаданных корреляционного правила:
   - `id`: задайте уникальный идентификатор правила.
   - `name`: задайте название правила.
   - `window_mode` (обязательное): определите режим работы корреляционного окна (`parallel`, `non-overlapping` или `stateless`).
   - `type` (обязательное): укажите `correlation_rule` в поле, чтобы определить тип правила.
   - `severity` (обязательное): задайте уровень угрозы испускаемого события. Доступны значения: `low`, `medium`, `high`, `critical`.
   - `version`: укажите версию правила в формате Semantic Versioning.
   - `author` (опциональное): укажите имя и контактные данные автора.
   - `description` (опциональное): предоставьте краткое описание назначения и функциональности правила.
   - `tags` (опциональное): добавьте теги для классификации и поиска правила.
2. Заполните общие поля корреляционного правила:
   - `filter` (опциональное): создайте блок правил на языке VRL, который определит, к каким событиям будет применяться правило.
   - `group_by` (опциональное): определите поля, по которым будет производиться группировка событий для корреляции.

- `on_correlation` (опциональное): создайте блок правил на языке VRL, которые необходимо выполнить после корреляции событий.
- `throttle_time_sec` (опциональное): установите интервал в секундах, который определяет, как часто могут генерироваться корреляционные события для каждой группы в поле `group_by`.

3. Заполните специфические поля в зависимости от выбранного режима корреляции (`window_mode`):

   Для правила `parallel`:

   - `start_when` (опциональное): создайте блок правил на языке VRL, определяющий условие открытия корреляционного окна. Если условие не задано, новое окно будет открываться на каждое событие прошедшее через фильтр.
   - `on_event` (обязательное): создайте блок правил на языке VRL, определяющий логику обновления состояния окна корреляции при получении событий. Если событие не обновляет окно, необходимо вызвать инструкцию `abort`.
   - `correlate_when` (обязательное): создайте блок правил на языке VRL, определяющий условие закрытия корреляционного окна и генерации корреляционного события. Если условие удовлетворено, окно будет закрыто, и по нему будет сформировано корреляционное событие.
   - `expire_after_sec` (обязательное): задайте период времени, по истечении которого корреляционное окно будет закрыто автоматически.
   - `windows_limit` (опциональное): ограничьте количество корреляционных окон, которые можно открыть правилом (по умолчанию 100 000).

   Для правила `non-overlapping`:

   - `on_event` (обязательное): создайте блок правил на языке VRL, определяющий логику обновления состояния окна корреляции при получении событий. Если событие не обновляет окно, необходимо вызвать инструкцию `abort`.
   - `correlate_when` (обязательное): создайте блок правил на языке VRL, определяющий условие закрытия корреляционного окна и генерации корреляционного события. Если условие удовлетворено, окно будет закрыто, и по нему будет сформировано корреляционное событие.
   - `expire_after_sec` (обязательное): задайте период времени, по истечении которого корреляционное окно будет закрыто автоматически.

   Для правила `stateless`:

   - `on_event` (опциональное): создайте блок правил на языке VRL, определяющий логику обновления состояния окна корреляции при получении событий. Если событие не обновляет окно, необходимо вызвать инструкцию `abort`.
   - `correlate_when` (обязательное): создайте блок правил на языке VRL, определяющий условие закрытия корреляционного окна и генерации корреляционного события. Если условие удовлетворено, окно будет закрыто, и по нему будет сформировано корреляционное событие.

4. Создайте блок условий тестирования работы правила корреляции в поле `tests`.

Тестирование правила

Тесты в правиле корреляции позволяют проверить корректность работы правила и его способность

реагировать на заданные типы событий.

Тестирование правила корреляции гарантирует, что правило корректно обрабатывает данные, точно идентифицирует события и правильно реагирует на них, обеспечивая надежную работу системы.

В случае, если правило корреляции использует данные из активных списков, необходимо инициализировать искусственные данные через поле `setup_tests`. Это позволит тестировать правило в условиях, максимально приближенных к реальной эксплуатации, без необходимости доступа к продуктивным данным.

Поле `setup_tests`

Для тестирования правил, которые используют внешние зависимости (например, взаимодействуют с активными списками или таблицами обогащения), использование поля `setup_tests` является обязательным.

Поле `setup_tests` в конфигурации правила корреляции предназначено для определения тестового окружения, включая предварительную настройку искусственных данных для активных списков и таблиц обогащения, используемых в правиле. Это обеспечивает проверку правила на корректность выполнения и взаимодействие с данными внешних зависимостей до его применения в операционной среде.

`setup_tests` используется для:

- Имитации взаимодействия правила с активными списками и таблицами обогащения.
- Проверки логики правила с учетом ожидаемых данных обогащения.

Процедура настройки `setup_tests`:

1. В корне конфигурации правила указывается секция `setup_tests`.
2. Внутри `setup_tests` через ключ `enrichment_tables` или `active_lists` задаются искусственные таблицы обогащения и активные списки, используемые правилом.
3. Для каждого искусственного элемента указывается его структура и набор искусственных данных, которые будут использоваться при тестировании.
4. Искусственные данные должны соответствовать ожидаемым данным из реальных таблиц обогащения/активных списков и позволять полноценно проверить работу правила.

Пример 46. Структура `setup_tests` для активного списка:

```
setup_tests:
  active_lists:
    <имя_списка>:
      <поле>:
        - [<значение_поля1>]
        - [<значение_поля2>]
```

Пример 47. Структура `setup_tests` для таблицы обогащения:

```
setup_tests:
```

```
enrichment_tables:
  <имя_таблицы>:
    fields:
      - <поле1>
      - <поле2>
    data:
      - [<значение1_поля1>, <значение1_поля2>]
      - [<значение2_поля1>, <значение2_поля2>]
```

Поле `tests`

Поле `tests` включает набор тестов для проверки и верификации правила корреляции. Эти тесты помогают убедиться, что правило работает корректно и соответствует ожидаемым результатам.

Чтобы создать тестовый сценарий, в поле `tests` определите следующие параметры:

- В `name` укажите название теста.
- В `events` создайте массив событий, которые будут использоваться для тестирования. Эти события должны имитировать поток реальных исходных событий, которые поступают на коррелятор.
- В `assertion` установите условие или набор условий, которые должны быть выполнены в результате обработки тестовых событий правилом. Это может включать проверку конкретных полей в итоговых агрегированных событиях или подсчет количества таких событий.

Тестовые события

Тестовые события должны имитировать данные, которые в реальности поступают на коррелятор. Они должны соответствовать формату и структуре событий, обрабатываемых правилом корреляции и содержать поля, которые используются в правиле.

Если в правиле настроена корреляция по полю с типом `DateTime`, например `timestamp`, то значение этого поля в `tests` необходимо задавать в формате `t'<DateTime-значение>'`. Например: `t'15.05.2024 06:28:15'`. Это важно при корреляции на отсутствие событий.

В противном случае при запуске тестов правила система обработает поле в виде строки и корреляция по дате-времени будет работать некорректно.

Условия проверки

Условия должны точно отражать ожидаемый результат применения правила к тестовым событиям. Это может быть проверка наличия определенных полей в событии, соответствие их значений ожидаемым или сравнение количества событий с предполагаемым исходом. Для более сложных правил могут потребоваться более сложные утверждения, которые могут включать логические операции и сравнения.

Пример:

```
tests:
```

```
  - name: "Тестирование обнаружения DDoS атаки"
    events:
      - { source_ip: "192.168.1.1", event_type: "traffic_anomaly", traffic_volume:
"10000" }
    assertion: !vrl |
        assert_eq!(.alert_type, "DDoS", "Ожидается обнаружение DDoS атаки")
```

Обогащение через внешние зависимости

Обогащение корреляционных событий представляет собой процесс дополнения событий, сгенерированных в ходе корреляции, дополнительными данными. В декларативных правилах корреляции можно настроить логику обогащения в полях `on_event` и `on_correlation`.

В рамках языка VRL обогащение корреляционных событий включает в себя:

- применение специфических функций и выражений для динамического добавления или модификации данных в событиях;
- извлечение информации из активных списков, которые могут предоставлять как статическую, так и динамическую информацию в зависимости от настроек схемы активного списка;
- взаимодействие с таблицами обогащения, которые предоставляют статическую информацию, например, сведения о геолокации, данные организаций, соответствующие спискам разрешенных и заблокированных ресурсов.

Активные списки

Активные списки представляют собой структуры данных, которые используются для хранения и управления как динамически изменяющимися наборами информации, так и статическими списками и таблицами, такими как IP-адреса, доменные имена, идентификаторы устройств или пользователей. В контексте императивных правил корреляции, поля `on_correlation` и `on_event` могут включать вызовы функций для взаимодействия с активными списками, используя язык VRL. Это позволяет добавлять, извлекать и удалять записи из активных списков на основе логики обработки корреляционных событий.

Функции для взаимодействия с активными списками:

Функция	Описание	Параметры
`add_active_record(list, key, value)`	Добавляет запись в активный список. Если запись с таким ключом уже существует, значение обновляется. В противном случае создается новая запись.	• `list`: название схемы активного списка, в который добавляется запись. • `key`: уникальный ключ для записи. • `value`: значение записи.
`get_active_record(list, key)`	Извлекает значение из активного списка по указанному ключу. Возвращает значение, ассоциированное с ключом, или `null`, если запись отсутствует.	• `list`: название схемы активного списка, из которого нужно извлечь запись. • `key`: уникальный ключ записи.

Функция	Описание	Параметры
`remove_active_record (list, key)`	Удаляет запись из активного списка. Если запись с таким ключом отсутствует, ничего не происходит.	• `list`: название схемы активного списка, из которого необходимо удалить запись. • `key`: уникальный ключ записи, которую необходимо удалить.

Пример 48. Пример использования в поле `on_correlation` правила корреляции:

```
on_correlation: !vrl |
  if .threat_level == "High" {
    add_active_record!("suspicious_ips", { "ip": .source_ip, "is_suspicious": true })
  }

  if .event_type == "authentication_failure" {
    failure_count = get_active_record!("auth_failure_counts", { "user_id": .user_id
}).failure_count || 0
    add_active_record!("auth_failure_counts", { "user_id": .user_id, {"failure_count":
failure_count + 1 })
  }

  if .event_type == "authentication_success" {
    remove_active_record("auth_failure_counts", { "user_id": .user_id })
  }
```

В этом примере:

- IP-адреса с высоким уровнем угрозы добавляются в активный список подозрительных IP-адресов.
- Считается количество неудачных попыток аутентификации для пользователей и обновляется соответствующая запись в активном списке.
- При успешной аутентификации запись пользователя удаляется из списка неудачных попыток аутентификации, что может свидетельствовать об окончании атаки или восстановлении доступа пользователя.

Пример 49. Правило корреляции с активным списком, схема RObject

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule

# Название правила.
name: DDoSDetectionRule

# Режим работы корреляционного окна.
window_mode: parallel

# Тип правила.
type: correlation_rule
```

```
# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика и взаимодействия с активным
списком подозрительных IP.

# Имя и контактные данные автора правила.
author: John Doe <johndoe@example.com>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .event_type == "HTTP_REQUEST" &&
  .source_ip != null &&
  .request_path != null

# Логика группировки событий.
group_by:
  - source_ip

# Время жизни корреляционного окна в секундах.
expire_after_sec: 600

# Частота генерации корреляционных событий в рамках каждой группы.
throttle_time_sec: 60

# Логика обновления состояния окна корреляции при получении событий.
on_event: !vrl |
  if int!(%state.request_count) >= 10 {
    abort
  }
  %state.request_count = int(%state.request_count) + 1

# Условие открытия корреляционного окна.
```

```
start_when: !vrl |
  .event_type == "HTTP_REQUEST" && .source_ip != null

# Условие закрытия корреляционного окна и генерации корреляционного события.
correlate_when: !vrl |
  int!(%state.request_count) >= 10

# Логика, применяемая после генерации корреляционного события.
on_correlation: !vrl |
  let is_suspicious = get_active_record("suspicious_ips", .source_ip)
  if is_suspicious != null {
    .message = "Обнаружена атака DDoS с источника: " + .source_ip + ". IP помечен как
подозрительный."
  } else {
    .message = "Обнаружена атака DDoS с источника: " + .source_ip + ". IP не найден в
списке подозрительных."
  }

# Инициализация искусственных данных для активных списков.
setup_tests:
  active_lists:
    suspicious_ips:
      - source_ip: "192.0.2.1"
        is_suspicious: true
      - source_ip: "198.51.100.10"
        is_suspicious: false

# Тесты для проверки работы правила.
tests:
  - name: "Detect DDoS from suspicious IP"
    events:
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
    assertion: !vrl |
      assert_eq!(.message, "Обнаружена атака DDoS с источника: 192.0.2.1. IP помечен
как подозрительный.")
   - name: "Detect DDoS from non-suspicious IP"
     events:
       - { "event_type": "HTTP_REQUEST", "source_ip": "198.51.100.10", "request_path":
"/api" }
     assertion: !vrl |
       assert_eq!(.message, "Обнаружена атака DDoS с источника: 198.51.100.10. IP не
найден в списке подозрительных.")
```

Логика правила описывается в блоках `on_event`, `start_when`, `correlate_when`, `on_correlation`, где:

- `on_event` учитывает количество HTTP-запросов от каждого источника IP.
- `start_when` инициирует корреляционное окно при поступлении HTTP-запроса.
- `correlate_when` активирует генерацию корреляционного события, когда количество запросов достигает порога в 10.

- `on_correlation` дополняет корреляционное событие информацией о том, был ли источник IP помечен как подозрительный в активном списке.

Секция `setup_tests` с искусственными данными для активного списка `suspicious_ips` позволяет проверить правило в условиях, максимально приближенных к реальной эксплуатации, без необходимости доступа к продуктивным данным. Это обеспечивает возможность тестирования правила на взаимодействие с активным списком.

Тесты `Detect DDoS from suspicious IP` и `Detect DDoS from non-suspicious IP` демонстрируют работу правила в различных условиях, проверяя его способность корректно реагировать на атаки с подозрительных и не подозрительных IP-адресов соответственно. Успешное прохождение этих тестов подтверждает, что правило корреляции обнаруживает DDoS-атаки и правильно взаимодействует с активными списками для обогащения корреляционных событий дополнительной информацией об источнике атаки.

Таблицы обогащения

Таблицы обогащения представляют собой структурированные наборы данных, используемые для добавления контекстной информации к событиям в процессе их корреляции. Это может включать данные о геолокации, информацию об устройствах, базы данных угроз и другие внешние источники данных. В контексте императивных правил корреляции, поля `on_correlation` и `on_event` могут включать вызовы функции VRL для запроса и извлечения данных из таблиц обогащения, тем самым обогащая события дополнительными деталями для более глубокого анализа.

Функции для взаимодействия с таблицами обогащения:

- `find_enrichment_table_records(table, condition, [select, case_sensitive])`: позволяет выполнить поиск записей в таблице обогащения, которые соответствуют заданным условиям. Возвращает массив записей, удовлетворяющих условиям поиска.
- `get_enrichment_table_record(table, condition, [select, case_sensitive])`: используется для извлечения одной записи из таблицы обогащения, которая точно соответствует заданному условию. В случае, если запись не найдена или найдено более одной записи, функция возвращает ошибку.

Пример использования в поле `on_correlation`:

```
on_correlation: !vrl |
  let geo_info = find_enrichment_table_records("geoip", {"ip": .source_ip}, select:
["country_name", "city_name"])
  if !geo_info.is_empty() {
    .source_country = geo_info[0].country_name
    .source_city = geo_info[0].city_name
  }

  let threat_info = get_enrichment_table_record("threat_intelligence", {"ip":
.source_ip}, select: ["category", "score"])
  if !is_null(threat_info) {
    .threat_category = threat_info.category
    .threat_score = threat_info.score
```

```
}
```

В этом примере:

- Выполняется поиск геолокационной информации по IP-адресу источника события в таблице `geoip`, и результаты добавляются к событию как `.source_country` и `.source_city`.
- Затем производится запрос к таблице `threat_intelligence` для извлечения информации о категории угрозы и оценке уровня угрозы, ассоциированных с IP-адресом источника. Эти данные добавляются к событию как `.threat_category` и `.threat_score`.

Тестирование правила с внешними зависимостями

Тестирование правил корреляции, которые обращаются к таблицам обогащения и/или активным спискам, требует предварительной настройки тестового окружения с помощью поля `setup_tests`, которое позволяет определить искусственные данные для внешнего элемента.

Поле `setup_tests` позволяет инициализировать необходимые данные для таблиц обогащения и активных списков перед запуском тестов правила корреляции. Это обеспечивает возможность проверить не только синтаксическую корректность правила, но и его логику обработки данных при взаимодействии с внешними элементами.

Процесс тестирования

1. Определение искусственных данных: для начала тестирования необходимо определить искусственные данные для всех активных списков и таблиц обогащения, с которым взаимодействует тестируемое правило. Эти данные должны быть репрезентативными и охватывать различные сценарии использования списка в правиле.
2. Конфигурация `setup_tests`: в корне конфигурации правила корреляции необходимо добавить секцию `setup_tests`, где указываются все используемые таблицы обогащения и активные списки с их искусственными данными. Имя каждого внешнего элемента в секции должно точно совпадать с именем, используемым в теле правила.
3. Заполнение искусственных данных: внутри внешнего элемента в секции `setup_tests` указывается список полей и соответствующих им искусственных данных. Данные задаются в формате, соответствующем ожидаемым запросам к активному списку и/или таблице обогащения в рамках теста.

Примеры:

Пример 50. Структура искусственных данных для активного списка:

```
setup_tests:
  active_lists:
    suspicious_ips:
      - source_ip: "192.168.1.1"
        is_suspicious: true
      - source_ip: "10.10.10.10"
        is_suspicious: false
    auth_failure_counts:
      - user_id: "user1"
        failure_count: 1
```

```
      - user_id: "user2"
        failure_count: 2
```

Пример 51. Структура искусственных данных для таблицы обогащения:

```
  enrichment_tables:
    geoip:
      fields:
        - ip
        - country
        - city
      data:
        - ["8.8.8.8", "US", "Mountain View"]
        - ["9.9.9.9", "GB", "London"]
```

Создание и выполнение тестов

После настройки искусственных данных следующим шагом является создание набора тестов, которые будут проверять взаимодействие правила корреляции с активными списками. Каждый тест должен включать в себя:

1. Описание теста (`name`): краткое описание цели теста.
2. События (`events`): список событий, которые будут подаваться на вход правила в процессе тестирования.
3. Утверждения (`assertions`): логика проверки результата работы правила на основе предоставленных событий и искусственных данных активных списков.

Пример теста:

```
tests:
  - name: "Test for Suspicious IP Detection"
    events:
      - { source_ip: "192.168.1.1", event_type: "login_attempt" }
    assertion: !vrl |
        let is_suspicious = get_active_record("suspicious_ips", .source_ip);
        assert_eq!(is_suspicious, true, "IP should be marked as suspicious")
  - name: "Authentication Failure Count Test"
    events:
      - { user_id: "user2", event_type: "login_failure" }
    assertion: !vrl |
        let failure_count = get_active_record("auth_failure_counts", .user_id);
        assert_eq!(failure_count, 2, "User2 should have 2 authentication failures")
```

В разделе событий (`events`) указываются события, которые подаются на вход правила корреляции для проверки его срабатывания. В первом тесте это попытка входа с IP-адреса, который должен быть помечен как подозрительный. Во втором тесте — неудачная попытка входа от пользователя, для которого уже зафиксированы подобные инциденты.

Утверждения (`assertions`) содержат логику проверки, выполняемую после обработки событий правилом. В первом тесте проверяется, что IP-адрес из события корректно идентифицирован как подозрительный на основе данных активного списка `suspicious_ips`. Во втором — что количество неудачных попыток входа для пользователя соответствует ожидаемому значению, основанному на данных активного списка `auth_failure_counts`.

Если утверждение (`assertion`) выполнено успешно, это означает, что правило корреляции корректно обрабатывает входные данные и правильно взаимодействует с активными списками. В противном случае, если результат не соответствует ожидаемому, это указывает на проблемы в логике правила или в его взаимодействии с активными списками.

- Убедитесь, что искусственные данные адекватно отражают разнообразие реальных сценариев, с которыми правило будет сталкиваться в продуктивной среде.
- Тщательно проверяйте соответствие имен активных списков и таблиц обогащения в `setup_tests` именам, используемым в правиле корреляции.
- Используйте результаты тестирования для уточнения и оптимизации логики правила корреляции и взаимодействия с внешним элементом.

Пример правила корреляции

Пример 52. Правило корреляции, схема RObject

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule

# Название правила.
name: DDoSDetectionRule

# Режим работы корреляционного окна.
window_mode: parallel

# Тип правила.
type: correlation_rule

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика

# Имя и контактные данные автора правила корреляции.
author: John Doe <johndoe@rvision.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
```

```
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .event_type == "HTTP_REQUEST" &&
  .source_ip != null &&
  .request_path != null

# Логика группировки событий.
group_by:
  - source_ip

# Время жизни корреляционного окна в секундах.
expire_after_sec: 600

# Частота генерации корреляционных событий в рамках каждой группы.
throttle_time_sec: 60

# Логика обновления состояния окна корреляции при получении событий.
on_event: !vrl |
  if int!(%state.request_count) >= 10 {
    abort
  }
  %state.request_count = int!(%state.request_count) + 1

# Условие открытия корреляционного окна.
start_when: !vrl |
  .event_type == "HTTP_REQUEST" && .source_ip != null

# Условие закрытия корреляционного окна и генерации корреляционного события.
correlate_when: !vrl |
  int!(%state.request_count) >= 10

# Логика, применяемая после генерации корреляционного события.
on_correlation: !vrl |
  .message = "Обнаружена атака DDoS с источника: " + string!(.event.source_ip)
  .additional_info = "Обнаружено " + string!(%state.request_count) + " запросов с
одного IP за короткий промежуток времени"

# Тесты для проверки работы правила.
tests:
  - name: DDoS Attack Detected
    events:
```

```
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.3", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.2", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.4", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
    assertion: !vrl |
      assert_eq!(.message, "Обнаружена атака DDoS с источника: 192.0.2.1")
```

Интерпретация

Правило корреляции `DDoSDetectionRule` разработано с целью выявления DDoS-атак на основе анализа входящего сетевого трафика.

Метаданные:

- `id`: `example/ddos_detection_rule` — уникальный идентификатор правила.
- `name`: `DDoSDetectionRule` — идентификационное название правила.
- `version`: `1.0.0` — версия правила.
- `author`: `John Doe <johndoe@rvision.ru>` — создатель или команда, отвечающая за правило.
- `description`: уточняет цель правила — обнаружение DDoS-атак, основанное на анализе трафика.
- `tags`: помечено как `ddos` и `security`, что подсказывает об основной функциональности и сфере применения правила.

Параметры корреляции:

- `window_mode`: `parallel` — режим корреляции, где события анализируются параллельно.
- `expire_after_sec`: `600` — время жизни окна корреляции в секундах (10 минут). Окно закроется не

сгенерировав корреляционное событие, если в течение указанного времени не произойдет обновления состояния.

- `throttle_time_sec`: `60` — интервал в секундах, который определяет, как часто могут генерироваться корреляционные события для каждой группы в поле `group_by`.
- `type`: `correlation_rule` — тип правила.
- `severity`: `high` — уровень серьезности, присеваемый правилу.

Логика правила:

- `filter`: правило активируется только для событий с типом `HTTP_REQUEST`, имеющих определенный `source_ip` и `request_path`.
- `group_by`: события группируются по полю `source_ip`, что позволяет отслеживать активность каждого отдельного IP-адреса.
- `on_event`: если количество запросов (`request_count`) для данного источника IP достигает или превышает 10, то выполнение правила прерывается с помощью директивы `abort`. Это означает, что атака DDoS уже обнаружена, и состояние перестает меняться. В противном случае значение `request_count` увеличивается на 1.
- `start_when`: определяет условие, при котором начинается новое корреляционное окно. В данном случае это любое событие `HTTP_REQUEST` с непустым полем `source_ip`.
- `correlate_when`: корреляционное событие генерируется, когда количество запросов от одного IP достигает 10.
- `on_correlation`: после обнаружения атаки DDoS формируется корреляционное событие со следующими аттрибутами:
  - `type`: "correlation_rule",
  - `severity`: "high",
  - `event_type`: "DDoS Detection",
  - `source_ip`: "192.168.1.1",
  - `datetime`: "2023-09-28T12:45:00Z",
  - `message`: "Обнаружена атака DDoS с источника: 192.168.1.1",
  - `additional_info`: "Обнаружено 10 запросов с одного IP за короткий промежуток времени."

Тестирование правила:

Тест `DDoS Attack Detected` имитирует поток событий, включая 13 HTTP-запросов. Из них 10 запросов исходят от одного IP-адреса "192.168.1.1". После обработки этого потока событий правило должно генерировать одно корреляционное событие с сообщением о DDoS-атаке от IP-адреса "192.168.1.1".

Таким образом, тест `DDoS Attack Detected` проверяет, что правило корреляции правильно реагирует на атаку DDoS, а именно, что оно обнаруживает атаку, когда количество запросов от одного IP-адреса достигает 10 и более.

Таким образом, правило корреляции `DDoSDetectionRule` предназначено для отслеживания атаки DDoS на основе количества HTTP-запросов от одного источника IP. Как только количество запросов достигает 10 или более за короткий промежуток времени (10 минут), правило обнаруживает атаку и

генерирует соответствующее корреляционное событие.

Обработка событий правилом корреляции

Продолжим анализ работы правила `DDoSDetectionRule` на примере исходного события и его обработки.

Предположим, что в течение 10 минут на коррелятор, на котором установлено правило `DDoSDetectionRule`, поступило 10 JSON-событий с источника "192.168.1.1" и они имеют следующий вид:

Пример 53. Исходное событие, JSON

```
{
  "event_type": "HTTP_REQUEST",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:34:56Z",
  "request_path": "/api",
  "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/93.0.4472.124 Safari/537.36"
}
```

При обработке событий происходит срабатывание правила корреляции `DDoSDetectionRule`. Правило создает корреляционное окно каждый раз, когда событие проходит его фильтр. Как только для источника "192.168.1.1" накапливается 10 событий, правило генерирует корреляционное событие, которое будет иметь следующий вид:

Пример 54. Корреляционное событие, генерируемое правилом `DDoSDetectionRule`

```
{
  "type": "correlation_rule",
  "severity": "high",
  "event_type": "DDoS Detection",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:45:00Z",
  "message": "Обнаружена атака DDoS с источника: 192.168.1.1",
  "additional_info": "Обнаружено 10 запросов с одного IP за короткий промежуток
времени."
}
```

Таким образом, правило отслеживает аномально высокую активность отдельных IP-адресов на основе входящих HTTP-запросов.

Чтобы правило могло участвовать в обработке поступающих в систему событий, его необходимо добавить в коррелятор соответствующего коллектора.

Расширение конфигурации существующего правила

Рассмотрим сценарий, в котором необходимо расширить конфигурацию существующего правила корреляции DDoSDetectionRule с помощью дополнительного условия. Данное условие будет

учитывать количество HTTP-запросов, поступающих с одного источника, и ранжировать эти источники в зависимости от этого параметра. В зависимости от количества коррелируемых событий, корреляционному событию будет присвоен один из четырех уровней угрозы: "low" (низкий), "medium" (средний), "high" (высокий) или "critical" (критический).

▼ *Исходное правило корреляции "DDoSDetectionRule", схема RObject*

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule

# Название правила.
name: DDoSDetectionRule

# Режим работы корреляционного окна.
window_mode: parallel

# Тип правила.
type: correlation_rule

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика

# Имя и контактные данные автора правила корреляции.
author: John Doe <johndoe@rvision.ru>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .event_type == "HTTP_REQUEST" &&
  .source_ip != null &&
  .request_path != null

# Логика группировки событий.
```

```
group_by:
  - source_ip

# Время жизни корреляционного окна в секундах.
expire_after_sec: 600

# Частота генерации корреляционных событий в рамках каждой группы.
throttle_time_sec: 60

# Логика обновления состояния окна корреляции при получении событий.
on_event: !vrl |
  if int!(%state.request_count) >= 10 {
    abort
  }
  %state.request_count = int!(%state.request_count) + 1

# Условие открытия корреляционного окна.
start_when: !vrl |
  .event_type == "HTTP_REQUEST" && .source_ip != null

# Условие закрытия корреляционного окна и генерации корреляционного события.
correlate_when: !vrl |
  int!(%state.request_count) >= 10

# Логика, применяемая после генерации корреляционного события.
on_correlation: !vrl |
  .message = "Обнаружена атака DDoS с источника: " + string!(.event.source_ip)
  .additional_info = "Обнаружено " + string!(%state.request_count) + " запросов с
одного IP за короткий промежуток времени"

# Тесты для проверки работы правила.
tests:
  - name: DDoS Attack Detected
    events:
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.3", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.2", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
```

```
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.4", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
      - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path":
"/api" }
    assertion: !vrl |
      assert_eq!(.message, "Обнаружена атака DDoS с источника: 192.0.2.1")
```

Пример 55. Конфигурация расширенного правила, схема RObject

```
# Уникальный идентификатор правила.
id: example/enhanced_ddos_detection_rule

# Название правила.
name: EnhancedDDoSDetectionRule

# Режим работы корреляционного окна.
window_mode: parallel

# Тип правила.
type: correlation_rule

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Расширенное правило обнаружения атаки DDoS с классификацией уровней
угрозы

# Имя и контактные данные автора правила корреляции.
author: John Doe <johndoe@example.com>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
 - ddos
```

```
 - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .event_type == "HTTP_REQUEST" &&
  .source_ip != null &&
  .request_path != null

# Логика группировки событий.
group_by:
  - source_ip

# Время жизни корреляционного окна в секундах.
expire_after_sec: 600

# Частота генерации корреляционных событий в рамках каждой группы.
throttle_time_sec: 60

# Логика обновления состояния окна корреляции при получении событий.
on_event: !vrl |
  if int!(%state.request_count) >= 10 {
    abort
  }
  %state.request_count = int!(%state.request_count) + 1

# Условие открытия корреляционного окна.
start_when: !vrl |
  .event_type == "HTTP_REQUEST" && .source_ip != null

# Условие закрытия корреляционного окна и генерации корреляционного события.
correlate_when: !vrl |
  int!(%state.request_count) >= 10

# Логика, применяемая после генерации корреляционного события.
on_correlation: !vrl |
  .message = "Обнаружена атака DDoS с источника: " + .event.source_ip

  # Расширяем правило для классификации уровней угрозы в зависимости от количества
запросов
  .severity = select(
    int!(%state.request_count) >= 10 => "critical",
    int!(%state.request_count) >= 6 => "high",
    int!(%state.request_count) >= 3 => "medium",
    true => "low"
  )
  .alert_type = "DDoS Attack"
  .additional_info = "Обнаружено " + string!(%state.request_count) + " запросов с
одного IP за короткий промежуток времени"

# Тесты для проверки работы правила.
tests:
```

```
  - name: DDoS Attack Detected
    events:
     - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
     - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
     - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
     - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
     - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.3", "request_path": "/api"
}
     - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
     - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
     - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.2", "request_path": "/api"
}
     - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
     - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
     - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.4", "request_path": "/api"
}
     - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
     - { "event_type": "HTTP_REQUEST", "source_ip": "192.0.2.1", "request_path": "/api"
}
    assertion: !vrl |
      assert_eq!(.message, "Обнаружена атака DDoS с источника: 192.0.2.1")
```

Интерпретация

В поле `severity` расширенного правила устанавливается один из следующих уровней серьезности в зависимости от значения `%state.request_count`:

- Если значение `%state.request_count` больше или равно 10, то уровень серьезности будет установлен как `critical`.
- Если значение `%state.request_count` больше или равно 6, но меньше 10, то уровень серьезности будет установлен как `high`.
- Если значение `%state.request_count` больше или равно 3, но меньше 6, то уровень серьезности будет установлен как `medium`.
- В противном случае, если ни одно из вышеперечисленных условий не выполняется (`true`), то уровень серьезности будет установлен как `low`.

Предположим, что в течение 10 минут на правило `EnhancedDDoSDetectionRule` поступает 10 событий в формате JSON с источника "192.168.1.1", и они имеют следующий вид:

Пример 56. Исходное событие, JSON

```
{
  "event_type": "HTTP_REQUEST",
  "source_ip": "192.168.1.1",
  "datetime": "2023-08-15T12:34:56Z",
  "request_path": "/api/resource",
  "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/91.0.4472.124 Safari/537.36"
}
```

Тогда в результате срабатывании правила корреляции `EnhancedDDoSDetectionRule`, генерируемое корреляционное событие будет иметь следующий вид:

Пример 57. Корреляционное событие, генерируемое правилом `EnhancedDDoSDetectionRule`

```
{
  "type": "correlation_rule",
  "severity": "critical", # Уровень угрозы установлен в значение `critical` согласно
расширенному условию классификации уровней угрозы.
  "event_type": "DDoS Detection",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:45:00Z",
  "message": "Обнаружена атака DDoS с источника: 192.168.1.1",
}
```

Улучшенное правило `EnhancedDDoSDetectionRule` представляет собой расширенную версию исходного правила `DDoSDetectionRule`. Оно расширяет конфигурацию исходного правила так, чтобы при обнаружении DDoS-атак использовалось дополнительное условие, которое учитывает количество HTTP-запросов, приходящих от источников. Условие ранжирует источники по количеству запросов, чтобы присвоить событию один из 4 уровней угрозы: `low`, `medium`, `high` или `critical`. Таким образом, улучшенное правило классифицирует источники атаки DDoS на основе их активности.

Декларативные правила

В данном разделе представлен обзор структуры, принципа работы, процесса создания и расширения конфигурации декларативного правила корреляции. Управление корреляционными правилами в системе осуществляется в разделе **Экспертиза**.

О декларативных правилах корреляции

Декларативное правило корреляции в системе представляет собой структурированное описание желаемых корреляций между событиями безопасности, представленное в виде объекта данных в формате RObject (.ro). Это описание определяет условия корреляции на высоком уровне, фокусируясь на желаемом результате корреляции, а не на точной последовательности операций для его достижения.

Структура декларативного правила корреляции подразумевает определение псевдонимов (`aliases`) для событий и их взаимосвязей. Псевдонимы позволяют группировать события в отдельные потоки и связывать их в логические цепочки (`join`). Условия корреляции и агрегации задаются отдельно для

каждой группы событий, что позволяет системе объединять их в единые корреляционные события.

Процесс корреляции состоит из последовательности операций сопоставления, где каждая операция связывает события на основе определенных критериев. Корреляция завершается, когда событие проходит через всю цепочку операций и соответствует всем условиям. Если задана группировка, событие может быть агрегировано с другими по установленным критериям. Дополнительно, корреляционное событие, можно обогатить используя правила, написанные на языке описания трансформаций и корреляций VRL

Структура правила корреляции

Правила корреляции содержат набор обязательных и опциональных полей.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор правила корреляции.	строка	да
`name`	Название правила.	строка	да
`type`	Тип элемента. Всегда имеет значение `correlation_rule` для элементов экспертизы этого типа.	строка	да
`version`	Версия правила корреляции, представленная в формате Semantic Versioning.	строка	да
`author`	Имя и контактные данные автора правила корреляции.	строка	нет
`description`	Описание правила корреляции, содержащее дополнительную информацию о его назначении и функциональности.	строка	нет
`data_source`	Список источников данных, с которыми совместимо правило корреляции. Используется для классификации и быстрого поиска элементов экспертизы в системе.	массив строк массив объектов	нет
`status`	Статус разработки текущей версии правила. Может принимать любые текстовые значения. Примеры значений: • `debug` — версия в разработке; • `experimental` — экспериментальная версия; • `test` — версия для тестирования; • `stable` — стабильная версия.	строка	нет

Поле	Описание	Тип данных	Обязательное поле
`date`	Дата создания или изменения правила. Задается в формате `YYYY-MM-DD`, то есть "год-месяц-день". Поле не связано с полями **Дата создания** и **Дата изменения** в карточке элемента экспертизы.	строка	нет
`reference`	Список ссылок на дополнительную информацию об атаках, которые пытается обнаружить правило.	массив строк	нет
`known_false_positives`	Список возможных ложноположительных срабатываний правила. Позволяет определить, было ли правило отработано ложно или нет.	массив строк	нет
`severity`	Уровень угрозы корреляционного события, испускаемого правилом. Допустимые значения: • `critical` — критический; • `high` — высокий; • `medium` — средний; • `low` — низкий.	строка	да
`tags`	Список тегов, присвоенных правилу корреляции для классификации и быстрого поиска.	массив строк	нет
`rate_limit`	Параметры для ограничения генерации корреляционных событий. При превышении ограничений система приостанавливает работу правила корреляции.	объект	нет
`max_correlation_windows`	Максимально допустимое количество корреляционных окон. При превышении ограничения система останавливает работу правила корреляции.	число	нет
`filter`	Блок условий, определяющий события, к которым будет применяться правило корреляции. Условия задаются на языке VRL. При создании правила в конструкторе условия автоматически сохраняются с использованием синтаксиса !expression, но логика работы фильтра остается идентичной.	строка	нет

Поле	Описание	Тип данных	Обязательное поле
`aliases`	Псевдонимы потоков событий, которые используются для их идентификации в рамках правила.	объект	да
`select`	Определяет структуру корреляции, включая псевдонимы и условия в `join`. Указывает начальный псевдоним потока событий, с которого начинается процесс корреляции.	объект	да
`group`	Параметры для группировки результатов корреляции.	массив	нет
`ttl`	Максимальное общее время хранения промежуточных корреляционных данных в секундах для всей цепочки событий. При отсутствии индивидуального значения `wait_for` для конкретного `join`, используется значение из поля `ttl`. Является обязательным, если в правиле используются операции соединения (`join`) и группировки (`group`).	число	нет
`on_correlate`	Логика обогащения корреляционного события, выполняемая после успешного соответствия корреляционным условиям. Логика задается на языке VRL. При создании правила в конструкторе условия автоматически сохраняются с использованием синтаксиса !expression, но логика работы блока остается идентичной.	строка	нет
`setup_tests`	Настройка тестового окружения для правил, включая предварительную конфигурацию всех используемых в правиле внешних зависимостей, таких как активные списки. Это поле позволяет определить и инициализировать все необходимые ресурсы для выполнения тестов правила, включая искусственные данные для активных списков.	объект	нет (если нет внешних зависимостей)
`throttle_time_sec`	Указывает интервал в секундах, который определяет, как часто могут генерироваться корреляционные события для каждой группы в поле `group`.	число	нет
`tests`	Блок тестирования правила корреляции, содержащий массив тестовых событий и условия для проверки срабатывания правила.	объект	нет

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform> # Платформа или информационная система.
- <source> # Источник событий.
# События (EventId) или путь к журналу событий:
- <EventID_1>
- <EventID_2>
```

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `filter`

Поле	Описание	Тип данных	Обязательное поле
`filter`	Определяет условия для событий, к которым применяется правило корреляции. Условия задаются на языке VRL. Если правило создается через конструктор, условия сохраняются с использованием синтаксиса `!expression`, где выражения представляют собой вложенные структуры условий. При этом логика работы фильтра остается неизменной.	строка	нет

Синтаксис `filter` в режиме кода

Если правило создается в режиме кода, условия фильтрации задаются на языке VRL.

Пример 58. Пример фильтра, созданного в режиме кода

```
filter: !vrl |
  .sourceIp != null &&
  (.sourceIp == "192.168.0.1" || includes(["192.168.0.2"], .sourceIp)) &&
  .duser != "admin"
```

Синтаксис `filter` в режиме конструктора

При сохранении правила через конструктор фильтр автоматически записывается с использованием синтаксиса `!expression`, при этом логика работы фильтра остается неизменной. Такая структура описывает условия фильтрации в виде вложенных объектов.

Максимальный уровень вложенности для условий фильтрации, созданных в конструкторе, ограничен тремя уровнями. Это ограничение применимо для всех операторов, включая `$and`, `$or`, `$eq`, `$ne`, `$active_list` и `$enrichment_table`.

Если требуется создать более сложные условия, рекомендуется создавать RObject-конфигурацию правила корреляции вручную в режиме кода с использованием синтаксиса VRL.

Основные элементы синтаксиса:

- `$and`: логическое "И" для объединения нескольких условий.
- `$or`: логическое "ИЛИ" для объединения нескольких условий.
- `$eq`: условие равенства поля и значения.
- `$ne`: условие неравенства поля и значения.
- `$active_list`: проверка наличия значения в активном списке.
- `$enrichment_table`: проверка наличия значения в таблице обогащения.
- `$const`: задание фиксированного значения для использования в условиях.
- `$get`: извлечение значения из поля события по указанному пути.

Пример 59. Пример фильтра, созданного в конструкторе

```
filter: !expression |
  $and:
    - $or:
        - $eq:
            path: sourceIp
            value:
              $const: "192.168.0.1"
        - $active_list:
            key:
              - path: sourceIp
                value:
                  $const: "192.168.0.2"
            name: "SuspiciousIPs"
            operator: in
    - $ne:
        path: duser
        value:
          $const: "admin"
```

В этом примере:

- `$and`: объединяет два условия, которые должны одновременно выполняться.
- `$or`: объединяет два условия, одно из которых должно выполняться:
  - `$eq`: проверяет, что поле `sourceIp` равно `"192.168.0.1"`.
  - `$active_list`: проверяет, что `sourceIp` содержится в активном списке `SuspiciousIPs`.
- `$ne`: проверяет, что поле `duser` не равно `"admin"`.

Структура `rate_limit`

Поле	Описание	Тип данных	Обязательное поле
`max_rate`	Максимально допустимое количество корреляционных событий, генерируемых за `duration_secs` секунд.	число	да
`duration_secs`	Промежуток времени в секундах, в течение которого допустима генерация не более `max_rate` корреляционных событий.	число	да
`grace_period_secs`	Промежуток времени в секундах, в течение которого ожидается восстановление нормального режима работы правила. При нормальном режиме за `duration_secs` секунд правило генерирует не более `max_rate` корреляционных событий. Если при превышении `max_rate` за `duration_secs` количество событий не возвращается к нормальным значениям в течение периода `grace_period_secs`, правило корреляции отключается.	число	да

Структура `aliases`

Поле	Описание	Тип данных	Обязательное поле
Имя_псевдонима	Уникальное имя, идентифицирующее поток событий.	строка (ключ в объекте `aliases`)	да
`filter`	Условие на языке VRL, которое определяет критерии фильтрации событий.	строка	да

Структура `select`

Поле	Описание	Тип данных	Обязательное поле
`alias`	Определяет начальный псевдоним потока событий для корреляции.	строка	да

Поле	Описание	Тип данных	Обязательное поле
`join`	Описывает последовательность операций соединения других потоков событий с исходным, указанным в `alias`.	объект	да
`on`	Условия, по которым происходит соединение потоков событий.	массив	да
`absent`	Указывает, должно ли отсутствие события из потока `alias` рассматриваться как условие для перехода к следующему `join`. Если указано `absent: true`, то для этого `join` поле `wait_for` является обязательным. По умолчанию, если не указано, считается `false`.	булево	нет
`wait_for`	Время жизни для промежуточных результатов корреляции. Если не задано, используется значение из `ttl` верхнего уровня. Если указано `absent: true`, то `wait_for` определяет время ожидания для проверки условия `absent`.	число	нет (но обязательно, если `absent: true`)

Структура `join`

Поле	Описание	Тип данных	Обязательное поле
`join`	Определяет последующие псевдонимы, которые присоединяются к корреляции.	объект	да
`on`	Условия, по которым происходит соединение между псевдонимами потоков событий.	массив	да
`absent`	Указывает, должно ли отсутствие события из потока `alias` рассматриваться как условие для перехода к следующему `join`.	логический	нет
`wait_for`	Время жизни для промежуточных результатов корреляции. Если не задано, используется значение из `ttl` верхнего уровня. Если указано `absent: true`, то `wait_for` определяет время ожидания для проверки условия `absent`.	число	нет (но обязательно, если указано `absent: true`)

Структура `group`

Поле	Описание	Тип данных	Обязательное поле
`alias`	Идентифицирует псевдоним потока событий, по атрибутам которого будет производиться группировка.	строка	да

Поле	Описание	Тип данных	Обязательное поле
`by`	Поля, по которым будет производиться группировка.	массив строк	да
`count`	Количество элементов в группе.	число	да
`unique`	Указывает, должна ли группировка выполняться по уникальным значениям полей, указанных в `by`. По умолчанию, если не указано, считается `false`.	логический	нет

Структура `on_correlate`

Поле	Описание	Тип данных	Обязательное поле
`on_correlate`	Определяет логику обогащения корреляционного события, выполняемую после успешного соответствия условиям правила. Логика задается на языке VRL. Если правило создается через конструктор, она сохраняется с использованием синтаксиса `!expression`, при этом логика работы блока остается неизменной.	строка	нет

Синтаксис on_correlate в режиме кода

Если правило создается в режиме кода, логика обогащения задается на языке VRL.

Пример 60. Пример блока обогащения, созданного в режиме кода

```
on_correlate: !vrl |
  if .event_count > 50 {
    .alert_type = "DDoS"
  } else if .event_count > 10 {
    .alert_type = "brute_force"
  } else {
    .alert_type = "suspicious_activity"
  }

  .severity = "high"

  if exists(.event1.sourceIp) {
    .detected_ip = .event1.sourceIp
  } else {
    .detected_ip = "неизвестен"
  }

  if exists(.event2.duser) {
    .detected_user = .event2.duser
  } else {
```

```
    .detected_user = "неизвестен"
  }

  .message = "Обнаружена угроза типа " + .alert_type + " от IP-адреса " + .detected_ip
+
             ". Пользователь: " + .detected_user
```

В этом примере:

- `.event_count`: используется для определения типа угрозы.
  - Если значение больше 50, устанавливается `.alert_type` как `"DDoS"`.
  - Если значение больше 10, но не превышает 50, устанавливается `.alert_type` как `"brute_force"`.
  - В остальных случаях `.alert_type` устанавливается как `"suspicious_activity"`.
- `.severity`: статически задается как `"high"`.
- `.detected_ip`: извлекается из поля `.event1.sourceIp`. Если поле отсутствует, задается как `"неизвестен"`.
- `.detected_user`: извлекается из поля `.event2.duser`. Если поле отсутствует, задается как `"неизвестен"`.
- `.message`: формируется текстовое сообщение с указанием типа угрозы (`.alert_type`), IP-адреса (`.detected_ip`) и пользователя (`.detected_user`).

Синтаксис on_correlate в режиме конструктора

При сохранении правила через конструктор блок обогащения автоматически записывается с использованием синтаксиса `!expression`, при этом логика работы блока остается неизменной. В этом случае действия представляют собой последовательность вложенных объектов, описывающих логику обогащения. Структура сохраняется в формате, где каждое действие (`$assign`, `$vrl`, `$message`) и выражение (`$const`, `$get`) записываются в виде отдельных элементов.

Основные элементы синтаксиса:

- `$assign`: присваивает значение определенному полю в итоговом корреляционном событии. Значение может быть задано напрямую (`$const`) или извлечено из исходного события (`$get`).
- `$vrl`: выполняет программу, написанную на языке VRL, которая может использоваться для выполнения сложных вычислений, преобразований и проверки данных.
- `$message`: формирует текстовое сообщение, которое будет добавлено к корреляционному событию. Сообщение может использовать значения полей из событий или фиксированные строки.
- `$const`: используется для задания фиксированного значения, передаваемого в другие элементы, такие как `$assign`.
- `$get`: извлекает значение из поля или события. Поле задается через параметр `path`, а событие указывается через параметр `event`.

Пример 61. Пример блока обогащения, созданного в конструкторе

```
on_correlate:
  - $assign:
      path: "severity"
      value:
        $const: "high"
  - $assign:
      path: "detected_ip"
      value:
        $get:
          event: "event1"
          path: "sourceIp"
  - $assign:
      path: "detected_user"
      value:
        $get:
          event: "event2"
          path: "duser"
  - $vrl: |
      if .event_count > 50 {
        .alert_type = "DDoS"
      } else if .event_count > 10 {
        .alert_type = "brute_force"
      } else {
        .alert_type = "suspicious_activity"
      }
  - $message: '"Обнаружена угроза типа " + .alert_type + " от IP-адреса " +
.detected_ip +
               ". Пользователь: " + (.detected_user ?? "неизвестен")'
```

В этом примере:

- `$assign` используется для:
  - Назначения фиксированного значения `"high"` полю `severity`.
  - Получения и присвоения значения поля `sourceIp` из события `event1` в поле `detected_ip`.
  - Получения и присвоения значения поля `duser` из события `event2` в поле `detected_user`.
- `$vrl` применяется для выполнения программы на языке VRL:
  - В зависимости от значения `event_count` задается значение поля `alert_type`.
- `$message` используется для формирования текстового сообщения:
  - Сообщение включает тип угрозы, IP-адрес источника и имя пользователя (если доступно).

Структура `tests`

Поле	Описание	Тип данных	Обязательное поле
`name`	Название теста, позволяющее идентифицировать его в тестовой выдаче.	строка	да
`events`	Поток тестовых событий, обрабатываемых правилом.	массив	да
`assertion`	Блок правил на языке VRL, который будет вызываться на каждое событие, испускаемое полем `events`. Используя встроенные функции `assert` и `assert_eq`, можно строить проверки на соответствие испускаемого события тестовым условиям.	строка	да

Операторы

В декларативных правилах операторы используются для определения логики соединения и анализа данных между различными потоками событий. Они позволяют формулировать условия для сопоставления и группировки событий, основываясь на определенных атрибутах или характеристиках этих событий.

На данный момент в корреляционных правилах поддерживается только оператор `eq`.

Оператор `eq`

Оператор `eq` используется для определения условия равенства между полями в различных потоках событий. Этот оператор является частью механизма `join`, который позволяет связывать события на основе общих характеристик или атрибутов.

Синтаксис

Оператор `eq` используется внутри блока `join` и принимает следующий синтаксис:

```
join:
  alias: <alias_name>
  on:
    - eq: { <alias_1>: <field_1>, <alias_2>: <field_2> }
```

Здесь `<alias_name>` — это псевдоним потока событий, к которому применяется оператор `join`, `<alias_1>` и `<alias_2>` — это псевдонимы потоков событий, между которыми осуществляется сравнение, а `<field_1>` и `<field_2>` — это поля в этих потоках событий, которые сравниваются.

Пример:

В следующем примере демонстрируется использование оператора `eq` для соединения двух потоков событий на основе равенства IP-адресов источника:

```
aliases:
  Traffic1:
    filter: !vrl |
      .event_type == "HTTP_REQUEST"
  Traffic2:
    filter: !vrl |
      .event_type == "FTP_REQUEST"

select:
  alias: Traffic1
  join:
    alias: Traffic2
    on:
      - eq: { Traffic1: .source_ip, Traffic2: .source_ip }
```

В этом примере, события из `Traffic1` и `Traffic2` соединяются, когда поле `.source_ip` совпадает между этими двумя потоками.

Составление правила

Управление корреляционными правилами в системе осуществляется в разделе **Экспертиза**.

Для составления правила корреляции в системе, необходимо заполнить обязательные поля правила. Это включает поля метаданных и поля, отвечающие за логику работы правила.

1. Определите метаданные правила:
    - `id`: задайте уникальный идентификатор правила.
    - `name`: задайте название правила.
    - `version`: укажите версию правила, следуя стандартам Semantic Versioning.
    - `type`: определите тип правила как `correlation_rule`.
    - `window_mode`: оставьте поле пустым, чтобы система ожидала декларативное правило.
    - `author` (опциональное): укажите имя и контактные данные автора.
    - `description` (опциональное): предоставьте краткое описание назначения и функциональности правила.
    - `data_source` (опциональное): укажите список совместимых с правилом источников данных для классификации правила в системе.
    - `status` (опционально): опишите статус разработки текущей версии правила.
    - `date` (опционально): укажите дату создания или изменения правила.
    - `reference` (опционально): предоставьте список ссылок на дополнительный материал об атаках, которые пытается обнаружить правило.
    - `known_false_positives` (опционально): укажите список возможных ложноположительных срабатываний правила.

- `severity`: укажите уровень угрозы корреляционного события, испускаемого правилом (`critical`, `high`, `medium` или `low`).
- `tags` (опциональное): добавьте теги для классификации правила.

2. Разработайте логику фильтрации и корреляции событий:
   - `filter`: создайте условие для фильтрации входящих событий. События, которые не удовлетворяют условиям фильтра, исключаются из дальнейшего анализа и не учитываются в цепочке корреляционных операций.
   - `aliases`: создайте псевдонимы для потоков событий с соответствующими фильтрами.
   - `select` и `join`: настройте логику соединения потоков событий, начиная с указанного псевдонима (`alias`) и определяя условия соединения (`on`).
   - `ttl`: определите общее максимальное время (`ttl`) для всей цепочки корреляционных операций, чтобы ограничить продолжительность хранения промежуточных результатов в памяти.
3. Задайте логику группировки событий (опционально):
   - `group`: определите параметры для группировки результатов корреляции, включая псевдоним (`alias`), поля для группировки (`by`) и количество элементов в группе (`count`).
   - `throttle_time_sec`: определите интервал в секундах, который определяет, как часто могут генерироваться корреляционные события для каждой группы в поле `group`.
4. Опишите пост-корреляционные действия (опционально):
   - `on_correlate`: определите логику обогащения корреляционного события, используя язык VRL. VRL-программа будет применяться к корреляционному событию после успешной корреляции.
5. Разработайте тесты для правила (опционально):
   - `tests`: создайте тесты, включающие различные типы событий и условия, для проверки эффективности и точности правила корреляции.

Поля метаданных

Метаданные являются основой для идентификации и описания правила корреляции в системе. Они должны быть заполнены следующим образом:

- `id`: установите уникальный идентификатор правила. Этот идентификатор должен быть уникальным в рамках системы.
- `name`: укажите название правила.
- `version`: задайте версию правила, следуя принципам Semantic Versioning (например, 1.0.0).
- `type`: установите тип правила `correlation_rule`.
- `author`: укажите имя и контактные данные создателя правила.
- `description`: предоставьте краткое описание назначения и функциональности правила.
- `data_source`: укажите список совместимых с правилом источников данных для классификации правила в системе.
- `status`: опишите статус разработки текущей версии правила.
- `date`: укажите дату создания или изменения правила.

- `reference`: предоставьте список ссылок на дополнительный материал об атаках, которые пытается обнаружить правило.
- `known_false_positives`: укажите список возможных ложноположительных срабатываний правила.
- `severity`: укажите уровень угрозы корреляционного события, испускаемого правилом (`critical`, `high`, `medium` или `low`).
- `tags`: укажите теги для классификации правила в системе.

Поле `rate_limit`

Поле `rate_limit` позволяет задать ограничения на генерацию корреляционных событий.

1. Укажите в поле `max_rate` максимально допустимое количество корреляционных событий, которое должно генерировать правило.
2. Укажите в поле `duration_secs` период времени в секундах, в течение которого допускается генерация не более `max_rate` корреляционных событий.
3. Укажите в поле `grace_period_secs` период времени в секундах, в течение которого система будет ожидать восстановления нормальной работы правила.

Если за `duration_secs` секунд будет сгенерировано более `max_rate` событий, то система будет ожидать восстановления нормальной работы правила в течение `grace_period_secs` секунд. Если нормальная работа правила не восстановится, система отключит правило корреляции.

Ограничение работает только при отключенном режиме распределенной корреляции.

Пример:

```
rate_limit:
    max_rate: 100
    duration_secs: 60
    grace_period_secs: 300
```

В этом примере устанавливается ограничение на генерацию не более 100 корреляционных событий за 60 секунд (1 минуту). При превышении заданного ограничения, если в течение последующих 300 секунд (5 минут) все еще будет генерироваться более 100 событий в минуту, система отключит правило корреляции.

Поле `max_correlation_windows`

Поле `max_correlation_windows` определяет максимально допустимое количество корреляционных окон. При превышении система останавливает работу правила корреляции. Для возобновления работы правила необходимо заново установить конфигурацию конвейера.

Ограничение работает только при отключенном режиме распределенной корреляции.

Пример:

```
max_correlation_windows: 100
```

В этом примере устанавливается ограничение на открытие не более 100 корреляционных окон.

Поле `filter`

Поле `filter` определяет подмножества событий, которые будут обработаны правилом. Это поле использует язык VRL для описания логики фильтрации, позволяя точно настроить, какие события будут участвовать в корреляции.

Предназначение поля `filter` — оценить каждое входящее событие и определить, соответствует ли оно заданным критериям. Если событие не соответствует условиям фильтра, оно исключается из дальнейшего анализа и не учитывается в корреляционном анализе. Это позволяет снизить объем обрабатываемых данных и увеличить точность и эффективность корреляционных правил.

В VRL для написания условий фильтра используются логические операторы (`&&`, `||`, `!`), сравнения (`==`, `!=`, `<`, `>`, `<=`, `>=`) и функции языка для работы с данными (например, `includes`, `starts_with`). Это позволяет создавать сложные и гибкие условия фильтрации.

Пример:

```
filter: !vrl |
    .source_ip != null &&          # Событие должно иметь ненулевой IP-адрес источника
    .request_path != null &&       # Путь запроса также должен быть указан
    .response_code == 404          # Код ответа должен быть 404
```

В этом примере фильтр выбирает события, у которых есть IP-адрес источника и путь запроса, а также код ответа равен 404. Это может быть полезно для выявления потенциальных попыток несанкционированного доступа или сканирования уязвимостей на веб-сервере.

Использование глобальных функций

Если условия фильтра нужно использовать в нескольких правилах, вы можете создать глобальную функцию с необходимой логикой. Создадим для указанного примера глобальную функцию со следующим исполняемым кодом и аргументами:

```
...
# Имя функции.
name: check_request
...

# Основной исполняемый код.
source: !vrl |
  source_ip != null &&
  request_path != null &&
  response_code == 404
```

```
# Аргументы функции.
arguments:
  - keyword: source_ip
    kind: bytes
  - keyword: request_path
    kind: bytes
  - keyword: response_code
    kind: integer
...
```

Тогда в фильтрах можно вызывать глобальную функцию так:

```
filter: !vrl |
    check_request(.source_ip, .request_path, .response_code)
```

Поле `aliases`

Поле `aliases` в правиле корреляции используется для разделения и фильтрации потока событий.

1. Определите псевдонимы. Назначьте каждому потоку событий уникальный псевдоним. Это имя будет использоваться для ссылки на этот поток в других частях правила.

   Например, для потока неудачных попыток входа можно использовать псевдоним `FailedLogins`.

2. Сформулируйте фильтры для каждого псевдонима на языке VRL, аналогично полю `filter`. Укажите, какие события будут относиться к данному потоку, основываясь на их атрибутах, таких как тип события, источник или содержание.

Пример фильтра: `.event_type == "login_failure" && .source_ip != null` — этот фильтр отбирает все события неудачных попыток входа, где указан IP-адрес источника.

Пример:

```
aliases:
  HighRateTraffic:
    filter: !vrl |
      .event_type == "login_failure" && .source_ip != null
  SuspiciousPatterns:
    filter: !vrl |
      .event_type == "HTTP_REQUEST"
```

Поле `select`

Поле `select` в правилах корреляции определяет логику соединения потоков событий. Это поле обеспечивает структурирование и организацию процесса корреляции, позволяя связывать события из разных источников на основе заданных критериев. Левый псевдоним в `join` является основным источником событий, к которому добавляются события из правого псевдонима, удовлетворяющие условиям соединения. Если в правом псевдониме нет событий, соответствующих условиям соединения, то события из левого псевдонима все равно будут обработаны, но без добавления

данных из правого псевдонима.

1. Выберите исходный псевдоним. Укажите в поле `alias` псевдоним, который будет использоваться как отправная точка для процесса корреляции. Этот псевдоним должен быть одним из определенных в разделе `aliases`.
2. Разработайте логику соединения (`join`). Используйте раздел `join`, чтобы определить, какие дополнительные потоки событий будут привлекаться для корреляции. В каждом `join` указывается псевдоним другого потока событий, который будет соединяться с исходным.
3. Определите условия соединения (`on`). В каждом блоке `join` укажите условия соединения в разделе `on`, которые определяют критерии соотношения событий из разных потоков.
4. Используйте `absent` (опционально) для определения логики корреляции, основанной на отсутствии события.
5. Укажите `wait_for` (опционально, но обязательно при `absent: true`) для установки времени ожидания для каждой операции соединения. Система будет накапливать события в течение указанного времени для каждого `join` и затем анализировать их на предмет соответствия заданным критериям соединения. Если `absent: true`, наряду с поступлением событий система будет регистрировать и их отсутствие.

Пример:

```
select:
  alias: TrafficA
  join:
    alias: TrafficB
    on:
      - eq: { TrafficA: .source_ip, TrafficB: .source_ip }
    wait_for: 80
    join:
      alias: TrafficC
      absent: true
      on:
        - eq: { TrafficA: .source_ip, TrafficC: .source_ip }
      wait_for: 60
```

В этом примере, начиная с потока `TrafficA`, сначала происходит соединение с `TrafficB` по совпадению источников IP. Затем проверяется отсутствие событий из `TrafficC`, соответствующих тому же условию, в течение 60 секунд.

Можно создать цепочку `join`, соединяя более двух потоков событий. Это позволяет формировать сложные условия корреляции для выявления более сложных угроз или аномалий.

Поле `join`

Поле `join` в правиле корреляции отвечает за создание корреляционных отношений между потоками событий. Оно используется для объединения двух или более потоков событий на основе заданных критериев, включая временной интервал хранения промежуточных результатов корреляции (`ttl`).

- Структура `join`:
  - Каждый `join` включает поля `alias`, `on`, `absent` (опционально), и `wait_for` (опционально).
  - `alias` в `join` указывает псевдоним потока событий, который будет соединяться с начальным потоком, определенным в `select`.
  - `on` описывает условие соединения, обычно основываясь на совпадении значений определенных полей между событиями.
  - `absent` (опционально) определяет, должно ли отсутствие события из потока `alias` рассматриваться как условие для перехода к следующему `join`. Если указано `absent: true`, то для этого `join` поле `wait_for` является обязательным.
  - `wait_for` (опционально, если `absent: false`) устанавливает максимальное время хранения промежуточных результатов корреляции для конкретной операции соединения. Если для конкретного `join` не указано значение `wait_for`, применяется общее ограничение времени, заданное на уровне правила в поле `ttl`. `wait_for` является обязательным полем, если указано `absent: true`.
- Условия соединения `on`:
  - Условие `on` с операцией `eq` (равенство) используется для указания полей, по которым будет происходить сопоставление между потоками событий.
- Создание цепочек `join`:
  - В `join` можно формировать цепочки, где каждый последующий `join` добавляет новый поток событий к уже существующей группе.
  - Это позволяет строить многоуровневые корреляции, где каждый новый уровень добавляет уточнения или расширяет условия корреляции.

Пример:

Для соединения двух потоков событий `TrafficA` и `TrafficB`, когда `source_ip` в `TrafficA` равен `destination_ip` в `TrafficB`, и одновременно проверяется отсутствие события `TrafficC` в заданный временной интервал, используется следующая конструкция:

```
select:
  alias: TrafficA
  join:
    alias: TrafficB
    on:
      - eq: { TrafficA: .source_ip, TrafficB: .destination_ip }
    wait_for: 300  # Время жизни промежуточных результатов в секундах.
    join:
      alias: TrafficC
      absent: true
      on:
        - eq: { TrafficA: .source_ip, TrafficC: .destination_ip }
      wait_for: 60  # Время ожидания отсутствия события в секундах.
```

Поле `group`

Поле `group` в правиле корреляции отвечает за группировку событий, что позволяет проводить более сложный и точный корреляционный анализ.

1. Определите псевдоним для группировки. В `alias` указывается псевдоним потока событий, для которого может быть произведена группировка. Этот псевдоним должен соответствовать одному из определенных в разделе `aliases`.
2. Выберите поля для группировки. Поле `by` определяет, по каким критериям (полям) будут группироваться события. Можно указать одно или несколько полей, на основе которых будут сформированы группы событий.
3. Установите лимит количества событий в группе. Поле `count` определяет минимальное количество событий в каждой группе. Этот параметр позволяет контролировать, когда группа достигает достаточного размера для корреляции или инициирования действия.
4. При необходимости определите признак уникальности группировки событий. В `unique` указывается, должна ли группировка выполняться по уникальным значениям полей, выбранных в `by`:
    - Если `unique: true`, то в группу будут собираться события с различными значениями полей.
    - Если `unique: false`, то в группу могут собираться события с одинаковыми значениями полей.

Пример:

```
group:
  - alias: TrafficA
    by:
      - source_ip
    count: 10
  - alias: TrafficB
    by:
      - suser
    count: 2
    unique: true
```

В данном примере производится группировка событий из потоков `TrafficA` и `TrafficB`:

- Для событий из потока `TrafficA` группировка производится по одинаковым значениям поля `source_ip`.
- Для событий из потока `TrafficB` группировка производится по уникальным значениям поля `suser`.

Корреляционное событие будет создано, когда в группе соберется 10 событий с одинаковым `source_ip` из потока `TrafficA` и 2 события с различными `suser` из потока `TrafficB`.

Можно задать несколько критериев в `by`, чтобы группировать события по более чем одному полю, что позволяет создавать более сложные условия корреляции.

Поле `throttle_time_sec`

Поле `throttle_time_sec` указывает интервал в секундах, который определяет, как часто могут генерироваться корреляционные события для каждой группы в поле `group`.

Пример:

```
throttle_time_sec: 60
```

В этом примере интервал между корреляционными событиями составляет 60 секунд для каждой группы.

Поле `ttl`

Поле `ttl` (time-to-live) определяет общее максимальное время в секундах, в течение которого промежуточные результаты всех корреляционных операций в правиле сохраняются в памяти, прежде чем будут удалены. Это значение применяется ко всем операциям корреляции в правиле. В отдельных `join` блоках может применяться поле `wait_for` для определения времени хранения промежуточных результатов корреляции конкретного `join` блока. При его отсутствии применяется общее значение `ttl` на уровне правила.

Если в правиле не используются операции соединения (`join`) и группировки (`group`), поле `ttl` является необязательным.

Пример:

```
select:
  alias: TrafficA
  join:
    alias: TrafficB
    on:
      - eq: { TrafficA: .source_ip, TrafficB: .destination_ip }
    ttl: 3600 # Промежуточные результаты для всей цепочки событий этого `join` будут
хранiться в течение 1 часа.
```

Поле `on_correlate`

Поле `on_correlate` в декларативных правилах корреляции используется для определения действий обогащения, которые выполняются после успешной корреляции событий. Эти действия могут включать преобразование данных, добавление новой информации к событиям или выполнение вычислений на основе данных событий. Для определения этих действий используется язык VRL. В действиях обогащения могут использоваться глобальные функции, извлечение информации из активных списков и взаимодействие с таблицами обогащения.

Глобальные функции не изменяют исходное событие и его метаданные.

Пример заполнения поля `on_correlate`:

```
on_correlate: !vrl |
  .message = "Обнаружена аномалия доступа с источника: " + .source_ip
  .threat_level = match .event_count {
    >= 1000 => "Критический",
    >= 500 => "Высокий",
    _ => "Средний"
  }
  .analysis_summary = "Источник " + .source_ip + " сгенерировал " +
tostring(.event_count) + " событий, что указывает на потенциальную угрозу."
```

В этом примере:

- Присваивается новое сообщение событию с детальным описанием обнаруженной аномалии.
- Устанавливается уровень угрозы на основе количества событий, связанных с источником.
- Формируется сводка анализа, включающая IP-адрес источника и общее количество событий.

Тестирование правила

Тесты в правиле корреляции позволяют проверить корректность работы правила и его способность реагировать на заданные типы событий.

Тестирование правила корреляции гарантирует, что правило корректно обрабатывает данные, точно идентифицирует события и правильно реагирует на них, обеспечивая надежную работу системы.

В случае, если правило корреляции использует глобальные функции или данные из активных списков и таблиц обогащения, необходимо инициализировать искусственные данные через поле `setup_tests`. Это позволит тестировать правило в условиях, максимально приближенных к реальной эксплуатации, без необходимости доступа к продуктивным данным.

Поле `setup_tests`

Для тестирования элементов экспертизы, которые используют внешние зависимости, то есть взаимодействуют с активными списками, таблицами обогащения или глобальными функциями, поле `setup_tests` является обязательным.

Поле `setup_tests` в конфигурации предназначено для определения тестового окружения, включая предварительную настройку искусственных данных для активных списков, таблиц обогащения и глобальных функций, используемых в элементе экспертизы. Это обеспечивает проверку правила на корректность выполнения и взаимодействие с внешними данными до его применения в операционной среде.

Поле `setup_tests` используется для:

- Имитации взаимодействия с активными списками, таблицами обогащения и глобальными функциями.
- Проверки логики правила или глобальной функции с учетом ожидаемых данных обогащения.

Процедура настройки `setup_tests`:

1. В корне конфигурации укажите секцию `setup_tests`.
2. Внутри секции `setup_tests` через ключ `enrichment_tables`, `active_lists` или `global_functions` задайте искусственные таблицы обогащения, активные списки и глобальные функции, используемые правилом.
3. Для каждого искусственного элемента укажите его структуру и набор искусственных данных, которые будут использоваться при тестировании.

   Искусственные данные должны соответствовать ожидаемым данным из реальных таблиц обогащения, активных списков или глобальных функций и позволять полноценно проверить работу элемента экспертизы.

Пример 62. Структура `setup_tests` для активного списка:

```
setup_tests:
  active_lists:
    <имя_списка>:
      <поле>:
        - [<значение_поля1>]
        - [<значение_поля2>]
```

Пример 63. Структура `setup_tests` для таблицы обогащения:

```
setup_tests:
  enrichment_tables:
    <имя_таблицы>:
      fields:
        - <поле1>
        - <поле2>
      data:
        - [<значение1_поля1>, <значение1_поля2>]
        - [<значение2_поля1>, <значение2_поля2>]
```

Пример 64. Структура `setup_tests` для глобальной функции:

```
setup_tests:
  global_functions:
    <имя_функции>:
      arguments:
        - keyword: <имя_параметра1>
          kind: <тип_параметра1>
        - keyword: <имя_параметра2>
          kind: <тип_параметра2>
      values_map:
        - argument: <аргументы1>
          event: <событие1>
          return: <результат1>
      default: <результат_по_умолчанию>
      fallible: <логическое значение>
```

При вызове искусственная глобальная функция возвращает значение, заданное в `values_map`, если найдена запись с совпадающими аргументами. Если подходящей записи нет, возвращается значение из поля `default`.

Поле `tests`

Поле `tests` включает набор тестов для проверки и верификации правила корреляции. Эти тесты помогают убедиться, что правило работает корректно и соответствует ожидаемым результатам.

Чтобы создать тестовый сценарий, в поле `tests` определите следующие параметры:

- В `name` укажите название теста.
- В `events` создайте массив событий, которые будут использоваться для тестирования. Эти события должны имитировать поток реальных исходных событий, которые поступают на коррелятор.
- В `assertion` установите условие или набор условий, которые должны быть выполнены в результате обработки тестовых событий правилом. Это может включать проверку конкретных полей в итоговых агрегированных событиях или подсчет количества таких событий.

Тестовые события

Тестовые события должны имитировать данные, которые в реальности поступают на коррелятор. Они должны соответствовать формату и структуре событий, обрабатываемых правилом корреляции и содержать поля, которые используются в правиле.

Если в правиле настроена корреляция по полю с типом `DateTime`, например `timestamp`, то значение этого поля в `tests` необходимо задавать в формате `t'<DateTime-значение>'`. Например: `t'15.05.2024 06:28:15'`. Это важно при корреляции на отсутствие событий.

В противном случае при запуске тестов правила система обработает поле в виде строки и корреляция по дате-времени будет работать некорректно.

Условия проверки

Условия должны точно отражать ожидаемый результат применения правила к тестовым событиям. Это может быть проверка наличия определенных полей в событии, соответствие их значений ожидаемым или сравнение количества событий с предполагаемым исходом. Для более сложных правил могут потребоваться более сложные утверждения, которые могут включать логические операции и сравнения.

Пример:

```
tests:
  - name: "Тестирование обнаружения DDoS атаки"
    events:
      - { source_ip: "192.168.1.1", event_type: "traffic_anomaly", traffic_volume:
"10000" }
    assertion: !vrl |
```

```
        assert_eq!(.alert_type, "DDoS", "Ожидается обнаружение DDoS атаки")
```

Обогащение через внешние зависимости

Обогащение корреляционных событий представляет собой процесс дополнения событий, сгенерированных в ходе корреляции, дополнительными данными. В декларативных правилах корреляции можно настроить логику обогащения в поле on_correlate.

В рамках языка VRL обогащение корреляционных событий включает в себя применение специфических функций и выражений для динамического добавления или модификации данных в событиях. Помимо этого, это может включать извлечение информации из активных списков, которые могут предоставлять как статическую, так и динамическую информацию в зависимости от настроек схемы активного списка. А также взаимодействие с таблицами обогащения, которые предоставляют статическую информацию, например, сведения о геолокации, данные организаций, соответствующие спискам разрешенных и заблокированных ресурсов.

Активные списки

Активные списки представляют собой структуры данных, которые используются для хранения и управления как динамически изменяющимися наборами информации, так и статическими списками и таблицами, такими как IP-адреса, доменные имена, идентификаторы устройств или пользователей. В контексте декларативных правил корреляции, поле on_correlate может включать вызовы функций для взаимодействия с активными списками, используя язык VRL. Это позволяет добавлять, извлекать и удалять записи из активных списков на основе логики обработки корреляционных событий.

Функции для взаимодействия с активными списками:

Функция	Описание	Параметры
`add_active_record(list, key, value)`	Добавляет запись в активный список. Если запись с таким ключом уже существует, значение обновляется. В противном случае создается новая запись.	• `list`: название схемы активного списка, в который добавляется запись. • `key`: уникальный ключ для записи. • `value`: значение записи.
`get_active_record(list, key)`	Извлекает значение из активного списка по указанному ключу. Возвращает значение, ассоциированное с ключом, или `null`, если запись отсутствует.	• `list`: название схемы активного списка, из которого нужно извлечь запись. • `key`: уникальный ключ записи.
`remove_active_record(list, key)`	Удаляет запись из активного списка. Если запись с таким ключом отсутствует, ничего не происходит.	• `list`: название схемы активного списка, из которого необходимо удалить запись. • `key`: уникальный ключ записи, которую необходимо удалить.

Пример 65. Пример использования в поле `on_correlation` правила корреляции:

```
on_correlation: !vrl |
  if .threat_level == "High" {
    add_active_record!("suspicious_ips", { "ip": .source_ip, "is_suspicious": true })
  }

  if .event_type == "authentication_failure" {
    failure_count = get_active_record!("auth_failure_counts", { "user_id": .user_id
}).failure_count || 0
    add_active_record!("auth_failure_counts", { "user_id": .user_id, {"failure_count":
failure_count + 1 })
  }

  if .event_type == "authentication_success" {
    remove_active_record("auth_failure_counts", { "user_id": .user_id })
  }
```

В этом примере:

- IP-адреса с высоким уровнем угрозы добавляются в активный список подозрительных IP-адресов.
- Считается количество неудачных попыток аутентификации для пользователей и обновляется соответствующая запись в активном списке.
- При успешной аутентификации запись пользователя удаляется из списка неудачных попыток аутентификации, что может свидетельствовать об окончании атаки или восстановлении доступа пользователя.

Таблицы обогащения

Таблицы обогащения представляют собой структурированные наборы данных, используемые для добавления контекстной информации к событиям в процессе их корреляции. Это может включать данные о геолокации, информацию об устройствах, базы данных угроз и другие внешние источники данных. В поле `on_correlate` декларативных правил корреляции можно использовать специализированные функции VRL для запроса и извлечения данных из таблиц обогащения, тем самым обогащая события дополнительными деталями для более глубокого анализа.

Функции для взаимодействия с таблицами обогащения:

- `find_enrichment_table_records(table, condition, [select, case_sensitive])`: позволяет выполнить поиск записей в таблице обогащения, которые соответствуют заданным условиям. Возвращает массив записей, удовлетворяющих условиям поиска.
- `get_enrichment_table_record(table, condition, [select, case_sensitive])`: используется для извлечения одной записи из таблицы обогащения, которая точно соответствует заданному условию. В случае, если запись не найдена или найдено более одной записи, функция возвращает ошибку.

Пример использования в поле `on_correlate`:

```
on_correlate: !vrl |
  let geo_info = find_enrichment_table_records("geoip", {"ip": .source_ip}, select:
["country_name", "city_name"])
  if !geo_info.is_empty() {
    .source_country = geo_info[0].country_name
    .source_city = geo_info[0].city_name
  }

  let threat_info = get_enrichment_table_record("threat_intelligence", {"ip":
.source_ip}, select: ["category", "score"])
  if !is_null(threat_info) {
    .threat_category = threat_info.category
    .threat_score = threat_info.score
  }
```

В этом примере:

- Выполняется поиск геолокационной информации по IP-адресу источника события в таблице `geoip`, и результаты добавляются к событию как `.source_country` и `.source_city`.
- Затем производится запрос к таблице `threat_intelligence` для извлечения информации о категории угрозы и оценке уровня угрозы, ассоциированных с IP-адресом источника. Эти данные добавляются к событию как `.threat_category` и `.threat_score`.

Тестирование правила с внешними зависимостями

Тестирование правил корреляции, которые обращаются к таблицам обогащения и/или активным спискам, требует предварительной настройки тестового окружения с помощью поля `setup_tests`, которое позволяет определить искусственные данные для внешнего элемента.

Поле `setup_tests` позволяет инициализировать необходимые данные для таблиц обогащения и активных списков перед запуском тестов правила корреляции. Это обеспечивает возможность проверить не только синтаксическую корректность правила, но и его логику обработки данных при взаимодействии с внешними элементами.

Процесс тестирования

1. Определение искусственных данных: для начала тестирования необходимо определить искусственные данные для всех активных списков и таблиц обогащения, с которым взаимодействует тестируемое правило. Эти данные должны быть репрезентативными и охватывать различные сценарии использования списка в правиле.
2. Конфигурация `setup_tests`: в корне конфигурации правила корреляции необходимо добавить секцию `setup_tests`, где указываются все используемые таблицы обогащения и активные списки с их искусственными данными. Имя каждого внешнего элемента в секции должно точно совпадать с именем, используемым в теле правила.
3. Заполнение искусственных данных: внутри внешнего элемента в секции `setup_tests` указывается список полей и соответствующих им искусственных данных. Данные задаются в формате, соответствующем ожидаемым запросам к активному списку и/или таблице обогащения в рамках теста.

Примеры:

Пример 66. Структура искусственных данных для активного списка:

```
setup_tests:
  active_lists:
    suspicious_ips:
      schema:
        source_ip: string
        is_suspicious: bool
      key:
        - ip
      data:
        - source_ip: "192.0.2.1"
          is_suspicious: true
        - source_ip: "192.0.2.2"
          is_suspicious: false
    auth_failure_counts:
      schema:
         user_id: string
         failure_count: integer
      key:
        - user_id
      data:
        - user_id: "user1"
          failure_count: 1
        - user_id: "user2"
          failure_count: 2
```

Пример 67. Структура искусственных данных для таблицы обогащения:

```
  enrichment_tables:
    geoip:
      fields:
        - ip
        - country
        - city
      data:
        - ["8.8.8.8", "US", "Mountain View"]
        - ["9.9.9.9", "GB", "London"]
```

Создание и выполнение тестов

После настройки искусственных данных следующим шагом является создание набора тестов, которые будут проверять взаимодействие правила корреляции с активными списками. Каждый тест должен включать в себя:

1. Описание теста (`name`): краткое описание цели теста.
2. События (`events`): список событий, которые будут подаваться на вход правила в процессе тестирования.

3. Утверждения (`assertions`): логика проверки результата работы правила на основе предоставленных событий и искусственных данных активных списков.

Пример теста:

```
tests:
  - name: "Test for Suspicious IP Detection"
    events:
      - { source_ip: "192.168.1.1", event_type: "login_attempt" }
    assertion: !vrl |
        is_suspicious = get_active_record!("suspicious_ips", {"ip":
.source_ip}.is_suspicious);
        assert_eq!(is_suspicious, true, "IP should be marked as suspicious")
  - name: "Authentication Failure Count Test"
    events:
      - { user_id: "user2", event_type: "login_failure" }
    assertion: !vrl |
        failure_count = get_active_record!("auth_failure_counts", {"user_id":
.user_id});
        assert_eq!(failure_count, 2, "User2 should have 2 authentication failures")
```

В разделе событий (`events`) указываются события, которые подаются на вход правила корреляции для проверки его срабатывания. В первом тесте это попытка входа с IP-адреса, который должен быть помечен как подозрительный. Во втором тесте — неудачная попытка входа от пользователя, для которого уже зафиксированы подобные инциденты.

Утверждения (`assertions`) содержат логику проверки, выполняемую после обработки событий правилом. В первом тесте проверяется, что IP-адрес из события корректно идентифицирован как подозрительный на основе данных активного списка `suspicious_ips`. Во втором — что количество неудачных попыток входа для пользователя соответствует ожидаемому значению, основанному на данных активного списка `auth_failure_counts`.

Если утверждение (`assertion`) выполнено успешно, это означает, что правило корреляции корректно обрабатывает входные данные и правильно взаимодействует с активными списками. В противном случае, если результат не соответствует ожидаемому, это указывает на проблемы в логике правила или в его взаимодействии с активными списками.

- Убедитесь, что искусственные данные адекватно отражают разнообразие реальных сценариев, с которыми правило будет сталкиваться в продуктивной среде.
- Тщательно проверяйте соответствие имен активных списков и таблиц обогащения в `setup_tests` именам, используемым в правиле корреляции.
- Используйте результаты тестирования для уточнения и оптимизации логики правила корреляции и взаимодействия с внешним элементом.

Пример:

▼ *Пример правила корреляции с активным списком, схема RObject*

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule_enrich

# Название правила.
name: DDoSDetectionRuleEnrich

# Тип элемента экспертизы.
type: correlation_rule

# Версия правила.
version: 1.0.0

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика с учетом репутации IP-адресов
из активного списка DDoSAttackIPs.

# Имя и контактные данные автора правила.
author: John Doe <johndoe@example.com>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .source_ip != null &&
  .request_path != null

# Разделение потоков событий на псевдонимы.
aliases:
  HighRateTraffic:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && parse_duration!(.window, "s") >= 100
  SuspiciousPatterns:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && ends_with(to_string!(.request_path),
"/api/highload")
  NormalTraffic:
```

```
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && ends_with(to_string!(.request_path), "/home")

# Соединение потоков событий.
select:
  alias: HighRateTraffic
  join:
    alias: SuspiciousPatterns
    absent: true
    on:
      - eq: { HighRateTraffic: .source_ip, SuspiciousPatterns: .source_ip }
    wait_for: 30

# Группировка событий.
group:
  - alias: HighRateTraffic
    by:
      - source_ip
    count: 3

# Частота генерации корреляционных событий в рамках каждой группы (событий в
секунду).
throttle_time_sec: 60

# Время хранения промежуточных корреляционных данных (секунды).
ttl: 60

# Логика обогащения корреляционного события на языке VRL.
on_correlate: !vrl |
  attack_info = get_active_record!("DDoSAttackIPs", {"ip":
%HighRateTraffic[0].source_ip});
  if attack_info != null {
    .attack_type = attack_info.attack_type
    .last_detected = attack_info.last_detected
    .attack_volume = attack_info.attack_volume
    .reputation_score = attack_info.reputation_score
    .involved_attacks = attack_info.involved_attacks
    .notes = attack_info.notes
  }
  .message = "Обнаружена потенциальная DDoS атака с источника: " +
to_string!(%HighRateTraffic[0].source_ip) +
              ", тип атаки: " + to_string!(.attack_type) +
              ", объем атаки: " + to_string!(.attack_volume) +
              ", репутация IP: " + to_string!(.reputation_score) +
              ", количество атак: " + to_string!(.involved_attacks) +
              ", заметки: " + to_string!(.notes)

  if .reputation_score != null && to_int!(.reputation_score) < 50 {
    .ddos_alert = "Высокий уровень запросов и низкая репутация IP, потенциальная
DDoS атака"
  }
```

```
# Искусственные данные для активного списка в тестовом окружении.
setup_tests:
  active_lists:
    DDoSAttackIPs:
      schema:
        ip: string
        last_detected: string
        attack_type: string
        reputation_score: integer
        attack_volume: string
        involved_attacks: integer
        notes: string
      key:
        - ip
      data:
      - ip: "192.0.2.1"
        last_detected: "2023-01-01T12:00:00Z"
        attack_type: "SYN Flood"
        reputation_score: 10
        attack_volume: "1.5 Gbps"
        involved_attacks: 5
        notes: "Repeated attacks on multiple targets"
      - ip: "192.0.2.3"
        last_detected: "2023-01-03T18:45:00Z"
        attack_type: "Amplification"
        reputation_score: 15
        attack_volume: "2 Gbps"
        involved_attacks: 3
        notes: "DNS amplification attack"
      - ip: "192.0.2.4"
        last_detected: "2023-01-04T20:00:00Z"
        attack_type: "HTTP Flood"
        reputation_score: 5
        attack_volume: "750 Mbps"
        involved_attacks: 4
        notes: "Targeted at web services"

# Тесты для проверки работы правила.
tests:
  - name: HighRateTraffic from Malware IP
    events:
      - { "timestamp": "t'2025-01-19T20:10:44+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "120s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:45+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "130s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:46+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "140s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:11:30+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.111", "window": "10s", "request_path": "/api/highload" }
```

```
    assertion: !vrl |
      assert_eq!(.[0]ddos_alert, "Высокий уровень запросов и низкая репутация IP,
потенциальная DDoS атака", "Alert generation failed for malware IP")
  - name: HighRateTraffic from Clean IP
    events:
      - { "timestamp": "t'2025-01-19T20:10:44+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.2", "window": "120s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:45+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.2", "window": "130s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:46+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.2", "window": "140s", "request_path": "start" }

    assertion: !vrl |
      assert!(.[0]ddos_alert == null || .[0]ddos_alert == "", "Alert should not be
generated for clean IP")
```

Пример правила корреляции

Пример 68. Правило корреляции, схема RObject

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule_basic

# Название правила.
name: DDoSDetectionRule

# Тип элемента экспертизы.
type: correlation_rule

# Версия правила.
version: 1.0.0

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика

# Имя и контактные данные автора правила.
author: John Doe <johndoe@example.com>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
```

```
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .source_ip != null &&
  .request_path != null

# Разделение потоков событий на псевдонимы.
aliases:
  HighRateTraffic:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && parse_duration!(.window, "s") >= 100
  SuspiciousPatterns:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && ends_with(to_string!(.request_path),
"/api/highload")

# Соединение потоков событий.
select:
  alias: HighRateTraffic
  join:
    alias: SuspiciousPatterns
    absent: true
    on:
      - eq: { HighRateTraffic: .source_ip, SuspiciousPatterns: .source_ip }
    wait_for: 30

# Группировка событий.
group:
  - alias: HighRateTraffic
    by:
      - source_ip
    count: 3

# Частота генерации корреляционных событий в рамках каждой группы (событий в секунду).
throttle_time_sec: 60

# Время хранения промежуточных корреляционных данных (секунды).
ttl: 60

# Логика обогащения корреляционного события на языке VRL.
on_correlate: !vrl |
  .message = "Обнаружена потенциальная DDoS атака с источника: " +
to_string!(%HighRateTraffic[0].source_ip)
  .additional_info = "Высокая частота запросов или подозрительные паттерны в пути
запроса /api/highload"

# Тесты для проверки работы правила.
tests:
```

```
  - name: DDoS Attack Detected
    events:
      - { "timestamp": "t'2025-01-19T20:10:44+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "120s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:45+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "130s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:46+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "140s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:11:30+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.111", "window": "10s", "request_path": "/api/highload" }
    assertion: !vrl |
      assert_eq!(.[0]message, "Обнаружена потенциальная DDoS атака с источника:
192.0.2.1")
```

Интерпретация

Правило корреляции "DDoSDetectionRule" предназначено для выявления DDoS-атак на основе анализа потока поступающих событий.

Метаданные правила:

- `id`: уникальный идентификатор правила в системе.
- `name`: название правила.
- `type`: тип `correlation_rule` указывает на то, что это правило корреляции.
- `version`: версия правила.
- `description`: описывает цель правила — "Обнаружение атаки DDoS на основе трафика".
- `author`: создатель правила.
- `tags`: теги `ddos` и `security` классифицируют правило по его функциональности и области применения.

Логика фильтрации (filter):

- Условия фильтрации `.source_ip != null` и `.request_path != null` выбирают события, у которых есть IP-адрес источника и путь запроса.

Логика фильтрации (aliases):

- Псевдоним `HighRateTraffic`: фильтр для событий с высокой частотой запросов, идентифицируемых по `event_type == "HTTP_REQUEST"` и `parse_duration!(.window) >= 100`. Отбирает события с частотой запросов, превышающей или равной 100 запросов в определенный временной интервал.
- Псевдоним `SuspiciousPatterns`: фильтр для событий с подозрительными путями запроса, определяемый как `event_type == "HTTP_REQUEST"` и `ends_with(to_string!(.request_path), "/api/highload")`. Отбирает запросы, пути которых оканчиваются на "/api/highload".

Соединение потоков событий (select и join):

- Выбор и соединение (`select` и `join`) потоков событий `HighRateTraffic` и `SuspiciousPatterns`

осуществляется на основе совпадения `source_ip`. Это соединение позволяет коррелировать высокочастотные запросы с подозрительными паттернами, исходящими от одного источника.

- В поле `wait_for` определяется, что система должна ожидать в течение 30 секунд перед тем, как принять решение о соединении событий из различных псевдонимов. Это значит, что система будет накапливать события в течение указанного времени с момента получения первого события и затем анализировать их на предмет соответствия заданным критериям соединения.
- В операции соединения с `SuspiciousPatterns` используется поле `absent: true`, что указывает на то, что отсутствие события из `SuspiciousPatterns` в течение 30 секунд будет добавлено в цепочку корреляционных событий.

Группировка событий (group):

- `group`: группировка событий применяется к псевдониму `HighRateTraffic` по полю `source_ip`, что позволяет агрегировать и анализировать события, исходящие от одного источника IP. Количество событий для формирования группы в поле `count` установлено в 3, что означает, что группа будет сформирована при накоплении 3 событий от одного источника IP.
- `throttle_time_sec`: задает интервал времени в секундах между генерацией корреляционных событий в рамках каждой группы событий. Например, 60 секунд.

Срок хранения промежуточных данных корреляции (ttl):

- `ttl` на уровне правила задает общее максимальное время жизни промежуточных корреляционных данных. Например, 60 секунд. Это время определяет, как долго промежуточные результаты для всей цепочки событий будут храниться в памяти до их автоматического удаления, если корреляция не завершается.

Пост-корреляционная логика (on_correlate):

- `on_correlate`: обогащает корреляционное событие полями `.message` и `.additional_info`.

По накоплении 3 событий правило генерирует корреляционное событие со следующими атрибутами:

- `type`: "correlation_rule",
- `event_type`: "DDoS Detection",
- `source_ip`: "192.168.1.1",
- `datetime`: "2023-09-28T12:45:00Z",
- `message`: "Обнаружена потенциальная DDoS атака с источника: 192.168.1.1",
- `additional_info`: "Высокая частота запросов и подозрительные паттерны в пути запроса /api/highload"

Тестирование правила (tests):

- Набор тестовых сценариев "DDoS Attack Detected" включает набор событий, часть из которых имитируют характеристики DDoS-атаки, а остальные являются контрольными примерами обычного трафика. Цель тестирования — подтвердить, что правило корректно идентифицирует только события, связанные с DDoS-атаками и игнорирует неподходящие события. Тесты включают события с HTTP-запросами, различающимися по `source_ip` и `request_path`, что позволяет проверить способность правила различать атакующий трафик от легитимного.

- События, моделирующие DDoS-атаку, имеют одинаковый `source_ip` и конечный путь запроса "/api/highload", что соответствует фильтрам `HighRateTraffic` и `SuspiciousPatterns`. Это сценарий, в котором ожидается активация правила и генерация корреляционного события.
- Контрольные события с разными `source_ip` и неподозрительными путями запроса ("/home", "/contact", "/about") предназначены для подтверждения, что правило не реагирует на обычные запросы, не связанные с DDoS-активностью.

`assertion` в тестовом блоке предназначен для проверки того, что только события, соответствующие характеристикам DDoS-атаки, приводят к формированию корреляционного события с ожидаемым сообщением об обнаружении атаки.

Обработка событий правилом корреляции

Проанализируем логику работы правила "DDoSDetectionRule", фокусируясь на процессе обработки событий.

Предположим, что за короткий промежуток времени на коррелятор, на котором установлено правило "DDoSDetectionRule", поступает поток JSON-событий от различных источников IP, включая множество запросов с IP-адреса "192.168.1.1". Эти события соответствуют критериям, заданным в фильтрах `HighRateTraffic` и `SuspiciousPatterns`.

Пример 69. Исходное событие, JSON

```
{
  "event_type": "HTTP_REQUEST",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:34:56Z",
  "request_path": "/api/highload",
  "user_agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/93.0.4472.124 Safari/537.36"
}
```

Здесь `x` в `source_ip` — уникальный номер для каждого события.

По мере поступления этих событий, они фильтруются правилом "DDoSDetectionRule". Фильтр `HighRateTraffic` отбирает события, соответствующие запросам с высокой частотой (более 100 запросов за заданный временной интервал), а `SuspiciousPatterns` фокусируется на запросах с определенным путем запроса `/api/highload`.

Когда события от одного источника IP, например, "192.168.1.1", соответствуют обоим фильтрам и проходят через механизм соединения `join`, правило "DDoSDetectionRule" приступает к группировке. Правило использует поле `group` для агрегирования событий от одного источника IP. Если количество событий от одного источника соответствует заданному порогу в 3 события, правило генерирует корреляционное событие.

Пример 70. Корреляционное событие, генерируемое правилом "DDoSDetectionRule"

```
{
  "type": "correlation_rule",
  "severity": "high",
```

```
  "event_type": "DDoS Detection",
  "source_ip": "192.168.1.1",
  "datetime": "2023-09-28T12:45:00Z",
  "message": "Обнаружена потенциальная DDoS атака с источника: 192.168.1.1",
  "additional_info": "Высокая частота запросов и подозрительные паттерны в пути
запроса /api/highload"
}
```

Данное корреляционное событие указывает на то, что система обнаружила аномальную активность, соответствующую критериям DDoS-атаки, исходящей с одного IP-адреса. Правило "DDoSDetectionRule" коррелирует события, основываясь на двух ключевых факторах: частоте запросов и специфических путях запросов, что является индикатором потенциальной атаки.

Расширение конфигурации существующего правила

Рассмотрим сценарий, в котором необходимо добавить дополнительные условия фильтрации и корреляции для правила "DDoSDetectionRule", чтобы повысить его эффективность и надежность в обнаружении более сложных DDoS-атак.

▼ *Исходное правило корреляции "DDoSDetectionRule", схема RObject*

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule_basic

# Название правила.
name: DDoSDetectionRule

# Тип элемента экспертизы.
type: correlation_rule

# Версия правила.
version: 1.0.0

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Описание назначения правила.
description: Обнаружение атаки DDoS на основе трафика

# Имя и контактные данные автора правила.
author: John Doe <johndoe@example.com>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1

# Теги, связанные с правилом.
```

```
tags:
  - ddos
  - security

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .source_ip != null &&
  .request_path != null

# Разделение потоков событий на псевдонимы.
aliases:
  HighRateTraffic:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && parse_duration!(.window, "s") >= 100
  SuspiciousPatterns:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && ends_with(to_string!(.request_path),
"/api/highload")

# Соединение потоков событий.
select:
  alias: HighRateTraffic
  join:
    alias: SuspiciousPatterns
    absent: true
    on:
      - eq: { HighRateTraffic: .source_ip, SuspiciousPatterns: .source_ip }
    wait_for: 30

# Группировка событий.
group:
  - alias: HighRateTraffic
    by:
      - source_ip
    count: 3

# Частота генерации корреляционных событий в рамках каждой группы (событий в
секунду).
throttle_time_sec: 60

# Время хранения промежуточных корреляционных данных (секунды).
ttl: 60

# Логика обогащения корреляционного события на языке VRL.
on_correlate: !vrl |
  .message = "Обнаружена потенциальная DDoS атака с источника: " +
to_string!(%HighRateTraffic[0].source_ip)
  .additional_info = "Высокая частота запросов или подозрительные паттерны в пути
запроса /api/highload"

# Тесты для проверки работы правила.
```

```
tests:
  - name: DDoS Attack Detected
    events:
      - { "timestamp": "t'2025-01-19T20:10:44+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "120s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:45+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "130s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:46+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "140s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:11:30+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.111", "window": "10s", "request_path": "/api/highload" }
    assertion: !vrl |
      assert_eq!(.[0]message, "Обнаружена потенциальная DDoS атака с источника:
192.0.2.1")
```

Для этого добавим дополнительный псевдоним с фильтрацией для отбора событий по значению поля `user_agent` исходного события. Также добавим обращение к активному списку для динамической оценки уровня угрозы, основанной на текущей информации о подозрительных IP-адресах.

Пример 71. Пример улучшенного правила "EnhancedDDoSDetectionRule", схема RObject

```
# Уникальный идентификатор правила.
id: example/ddos_detection_rule_enhanced

# Название правила.
name: EnhancedDDoSDetectionRule

# Тип элемента экспертизы.
type: correlation_rule

# Версия правила.
version: 1.0.0

# Уровень угрозы, присваиваемый корреляционному правилу.
severity: high

# Описание назначения правила.
description: Расширенное обнаружение DDoS-атак на основе трафика

# Имя и контактные данные автора правила.
author: John Doe <johndoe@example.com>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_13
    - EventID_1
```

```
# Теги, связанные с правилом.
tags:
  - ddos
  - security
  - enhanced_detection

# Фильтр применяется к событиям перед входом в правило.
filter: !vrl |
  .source_ip != null &&
  .request_path != null

# Разделение потоков событий на псевдонимы.
aliases:
  HighRateTraffic:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && parse_duration!(.window, "s") >= 100
  SuspiciousPatterns:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && ends_with(to_string!(.request_path),
"/api/highload")
  # Дополнительный псевдоним.
  UnusualUserAgents:
    filter: !vrl |
      .event_type == "HTTP_REQUEST" && !match(to_string!(.user_agent), r'Mozilla\/5\.0
\(Windows NT 10\.0; Win64; x64\) AppleWebKit\/537\.36 \(KHTML, like Gecko\)
Chrome\/91\.0\.4472\.124 Safari\/537\.36')

# Соединение потоков событий.
select:
  alias: HighRateTraffic
  join:
    alias: SuspiciousPatterns
    absent: true
    on:
      - eq: { HighRateTraffic: .source_ip, SuspiciousPatterns: .source_ip }
    wait_for: 30
    join:
      alias: UnusualUserAgents
      on:
        - eq: { HighRateTraffic: .user_agent, UnusualUserAgents: .user_agent }
      wait_for: 40

# Группировка событий.
group:
  - alias: HighRateTraffic
    by:
      - source_ip
      - user_agent # Дополнительный критерий фильтрации.
    count: 3
```

```
# Частота генерации корреляционных событий в рамках каждой группы (событий в секунду).
throttle_time_sec: 60

# Время хранения промежуточных корреляционных данных (секунды).
ttl: 60

# Логика обогащения корреляционного события на языке VRL.
on_correlate: !vrl |
  .message = "Обнаружена потенциальная DDoS атака с источника: " +
to_string!(%HighRateTraffic[0].source_ip)
  .additional_info = "Высокая частота запросов или подозрительные паттерны в пути
запроса /api/highload и необычные user agents"
  # Обращение к активному списку "suspicious_ips"
  if get_active_record!("suspicious_ips", {"ip": %HighRateTraffic[0].source_ip}) !=
null {
    .threat_level = "high"
  } else {
    .threat_level = "medium"
  }

# Искусственные данные активного списка.
setup_tests:
  active_lists:
    suspicious_ips:
      schema:
        ip: string
      key:
        - ip
      data:
        - ip: "192.0.2.1"
        - ip: "198.51.100.10"

# Тесты для проверки работы правила.
tests:
  - name: Enhanced DDoS Attack Detection
    events:
      - { "timestamp": "t'2025-01-19T20:10:44+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "120s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:45+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "130s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:10:46+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.1", "window": "140s", "request_path": "start" }
      - { "timestamp": "t'2025-01-19T20:11:30+00:00'", "event_type": "HTTP_REQUEST",
"source_ip": "192.0.2.111", "window": "10s", "request_path": "/api/highload" }
    assertion: !vrl |
      assert_eq!(.[0]message, "Обнаружена потенциальная DDoS атака с источника:
192.0.2.1")
      assert_eq!(.[0]threat_level, "high")
```

Таким образом, в улучшенное правило "EnhancedDDoSDetectionRule" внесены следующие изменения:

1. Дополнительный псевдоним и условие фильтрации:
    - Добавлен псевдоним `UnusualUserAgents`, в рамках которого происходит фильтрация событий по `user_agent` для идентификации необычных или подозрительных запросов.
    - Фильтр `.event_type == "HTTP_REQUEST" && regex_match("^Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (HTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36", .user_agent)` отбирает события со значением "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (HTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36" в поле `user_agent`.
    - Добавлен `UnusualUserAgents` в `select` с последующей корреляцией по `user_agent`.
2. Настройка `ttl`:
    - Значение `ttl` установлено в 80 секунд, увеличивая период хранения промежуточных корреляционных данных.
3. Дополнительный критерий группировки:
    - Добавлена группировка по `user_agent`, для более точного определения распределенных атак.
4. Интеграция с активным списком:
    - Использована функция `get_active_record("suspicious_ips", {"ip": %HighRateTraffic[0].source_ip})` для обогащения результатов проверки данными из активного списка "suspicious_ips". Это реализовано через условие:

      ```
        if get_active_record("suspicious_ips", {"ip": %HighRateTraffic[0].source_ip})
      != null {
          .threat_level = "high"
        } else {
          .threat_level = "medium"
        }
      ```

      Таким образом, при генерации корреляционного события, помимо основного сообщения и дополнительной информации, теперь также устанавливается уровень угрозы в зависимости от статуса IP-адреса в активных списках. Уровень угрозы устанавливается в "high", если IP-адрес источника обнаружен в списке подозрительных, и в "medium" в остальных случаях.

5. Тестовая проверка:
    - Добавлен блок `setup_tests` для предварительной настройки тестового окружения, включающий активный список `suspicious_ips` с указанными IP-адресами "192.0.2.1" и "198.51.100.10". Это позволяет эмулировать обращение к активному списку во время тестирования правила и проверять логику обогащения событий на основе наличия IP-адреса в списке подозрительных.
    - Добавлены тестовые события со значением "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (HTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36" в поле `user_agent`, чтобы проверить правило на способность корректно идентифицировать и обрабатывать события с конкретным значением в `user_agent`.
    - Добавлена проверка динамической оценки уровня угрозы на основе данных из активного списка. Проверяется, каким образом правило реагирует на обнаружение подозрительных IP-адресов и присваивает соответствующий уровень угрозы (`high` или `medium`).

Интерпретация

Основной поток (select):

- `alias: HighRateTraffic` определяет начальный поток событий для корреляции, который в данном случае представляет поток `HighRateTraffic`. Поток `HighRateTraffic` охватывает события, соответствующие определенным критериям, таким как высокая частота HTTP-запросов от одного источника IP.

Первый уровень соединения (join):

- `alias: SuspiciousPatterns` указывает на то, что поток `HighRateTraffic` будет соединяться с потоком `SuspiciousPatterns`. Поток `SuspiciousPatterns` фильтрует события на основе определенных паттернов в пути запроса (например, оканчивающихся на "/api/highload").
- `on: - eq: { HighRateTraffic: .source_ip, SuspiciousPatterns: .source_ip }` говорит о том, что соединение между потоками `HighRateTraffic` и `SuspiciousPatterns` будет осуществляться на основе совпадения значения поля `.source_ip`. То есть корреляция произойдет, если источник IP в обоих потоках событий совпадает.
- В операции соединения `HighRateTraffic` и `SuspiciousPatterns` используется поле `absent: true`, что указывает на то, что отсутствие события из `SuspiciousPatterns` в течение 30 секунд будет добавлено в цепочку корреляционных событий.

Второй уровень соединения (join):

- `alias: UnusualUserAgents` добавляет третий поток событий `UnusualUserAgents` в цепочку корреляции. Этот поток фокусируется на событиях с "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (HTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36" в поле `user_agent`.
- `on: - eq: { HighRateTraffic: .user_agent, UnusualUserAgents: .user_agent }` устанавливает, что соединение между `HighRateTraffic` и `UnusualUserAgents` осуществляется на основе совпадения по полю `.user_agent`. Это означает, что для корреляции в рамках данного правила событие должно соответствовать критериям как высокой частоты запросов, так и "подозрительного" `user_agent`.

Таким образом, создается многоуровневая логика корреляции, где события должны соответствовать критериям всех трех потоков — `HighRateTraffic`, `SuspiciousPatterns` и `UnusualUserAgents`. Это позволяет правилу "EnhancedDDoSDetectionRule" выявлять потенциальные DDoS-атаки, основываясь на более сложных и детализированных критериях, чем в исходной версии правила.

Правила нормализации

В данном разделе представлен обзор структуры, принципов работы, процесса создания и расширения конфигурации правила нормализации. Управление правилами нормализации в системе осуществляется в разделе **Экспертиза**.

О правилах нормализации

Правила нормализации в системе обрабатывают и преобразовывают поступающие события согласно условиям и коду нормализации, составленным на языке VRL. В данном разделе

представлен обзор процесса создания правила нормализации, его структуры и методов расширения конфигурации существующего правила нормализации.

Работа с правилом нормализации

Доступные операции над правилом нормализации:

- Создание правила нормализации
- Изменение правила нормализации
- Удаление правила нормализации

Создание правила нормализации

Чтобы добавить правило нормализации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило нормализации**. Система отобразит окно создания правила нормализации.

При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля правила нормализации, чтобы определить логику его работы.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст правило нормализации и отобразит соответствующее уведомление. Новое правило появится в таблице раздела **Экспертиза**.

Чтобы правило нормализации стало доступно на конвейерах событий, его необходимо включить.

Вы можете создать черновик правила нормализации без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик правила и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Изменение правила нормализации

Чтобы изменить правило нормализации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку правила нормализации в списке. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек правила.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

4. Внесите требуемые изменения в конфигурацию правила нормализации.
5. Нажмите на кнопку **Обновить версию**. Система опубликует новую версию правила нормализации с измененной конфигурацией и отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию правила нормализации в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Система сохранит конфигурацию для дальнейшего редактирования и отобразит соответствующее уведомление. В верхней части карточки правила будет отображаться предупреждение о наличии неопубликованных изменений.

Удаление правила нормализации

- Удаление доступно только для элементов экспертизы с типом создания **Пользовательский**.
- Если элемент экспертизы используется в других сущностях системы, его удаление невозможно.

Чтобы удалить правило нормализации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку правила нормализации в списке. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления правила.
4. Нажмите на кнопку **Удалить**. Система удалит правило нормализации и отобразит соответствующее уведомление. Удаленное правило будет исключено из списка элементов экспертизы.

Вы также можете удалить группу элементов экспертизы. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно удаления.

4. Нажмите на кнопку **Удалить**. Система удалит выбранные элементы и отобразит соответствующее уведомление. Удаленные элементы будет исключены из списка элементов экспертизы.

Структура правила нормализации

Правила нормализации описываются на языке RObject и содержат следующие поля:

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор правила.	строка	Да
`name`	Название правила, которое обозначает его цель или функцию.	строка	Да
`version`	Версия правила, представленная в формате Semantic Versioning, указывает на текущую версию правила.	строка	Да
`type`	Тип правила. Всегда имеет значение `normalization_rule` для правила этого типа.	строка	Да
`author`	Имя и контактные данные автора правила нормализации.	строка	Нет
`description`	Описание правила, где можно указать дополнительную информацию о его назначении и функциональности.	строка	Нет
`data_source`	Список источников данных, с которыми совместимо правило нормализации. Используется для классификации и быстрого поиска элементов экспертизы в системе.	массив строк массив объектов	нет
`tags`	Список тегов, присвоенных правилу нормализации для классификации и быстрого поиска.	массив строк	Нет
`filter`	Блок правил на языке VRL, определяющий условия (фильтр), при которых правило будет применяться к событиям. Фильтр позволяет выбирать определенные события для нормализации на основе заданных критериев.	строка	Нет
`normalizer`	Блок правил на языке VRL, определяющий правила нормализации события.	строка	Нет
`tests`	Блок тестирования правила нормализации, содержащий массив тестовых событий и условия для проверки корректности работы правила.	объект	Нет

Поле	Описание	Тип данных	Обязательное поле
`setup_tests`	Настройка тестового окружения для правил, включая предварительную конфигурацию всех используемых в правиле внешних зависимостей, таких как таблицы обогащения. Это поле позволяет определить и инициализировать все необходимые ресурсы для выполнения тестов правила, включая искусственные данные для активных списков.	объект	нет (если нет внешних зависимостей)

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform> # Платформа или информационная система.
- <source> # Источник событий.
# События (EventId) или путь к журналу событий:
- <EventID_1>
- <EventID_2>
```

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `tests`

Поле	Описание	Тип данных	Обязательное поле
`name`	Название теста, позволяющее идентифицировать его в тестовой выдаче.	строка	да
`events`	Поток тестовых событий, обрабатываемых правилом.	массив	да

Поле	Описание	Тип данных	Обязательное поле
`assertion`	Блок правил на языке VRL, который будет вызываться на каждое событие, испускаемое полем `events`. Используя встроенные функции `assert` и `assert_eq`, можно строить проверки на соответствие испускаемого события тестовым условиям.	строка	да

Структура `setup_tests`

Для тестирования элементов экспертизы, которые используют внешние зависимости, то есть взаимодействуют с активными списками, таблицами обогащения или глобальными функциями, поле `setup_tests` является обязательным.

Поле `setup_tests` в конфигурации предназначено для определения тестового окружения, включая предварительную настройку искусственных данных для активных списков, таблиц обогащения и глобальных функций, используемых в элементе экспертизы. Это обеспечивает проверку правила на корректность выполнения и взаимодействие с внешними данными до его применения в операционной среде.

Поле `setup_tests` используется для:

- Имитации взаимодействия с активными списками, таблицами обогащения и глобальными функциями.
- Проверки логики правила или глобальной функции с учетом ожидаемых данных обогащения.

Процедура настройки `setup_tests`:

1. В корне конфигурации укажите секцию `setup_tests`.
2. Внутри секции `setup_tests` через ключ `enrichment_tables`, `active_lists` или `global_functions` задайте искусственные таблицы обогащения, активные списки и глобальные функции, используемые правилом.
3. Для каждого искусственного элемента укажите его структуру и набор искусственных данных, которые будут использоваться при тестировании.

   Искусственные данные должны соответствовать ожидаемым данным из реальных таблиц обогащения, активных списков или глобальных функций и позволять полноценно проверить работу элемента экспертизы.

Пример 72. Структура `setup_tests` для активного списка:

```
setup_tests:
  active_lists:
    <имя_списка>:
      <поле>:
        - [<значение_поля1>]
```

```
        - [<значение_поля2>]
```

Пример 73. Структура `setup_tests` для таблицы обогащения:

```
setup_tests:
  enrichment_tables:
    <имя_таблицы>:
      fields:
        - <поле1>
        - <поле2>
      data:
        - [<значение1_поля1>, <значение1_поля2>]
        - [<значение2_поля1>, <значение2_поля2>]
```

Пример 74. Структура `setup_tests` для глобальной функции:

```
setup_tests:
  global_functions:
    <имя_функции>:
      arguments:
        - keyword: <имя_параметра1>
          kind: <тип_параметра1>
        - keyword: <имя_параметра2>
          kind: <тип_параметра2>
      values_map:
        - argument: <аргументы1>
          event: <событие1>
          return: <результат1>
      default: <результат_по_умолчанию>
      fallible: <логическое значение>
```

При вызове искусственная глобальная функция возвращает значение, заданное в `values_map`, если найдена запись с совпадающими аргументами. Если подходящей записи нет, возвращается значение из поля `default`.

Составление правила

Для составления правила нормализации в системе необходимо выполнить следующие шаги:

1. Определите метаданные правила:
    - `id`: задайте уникальный идентификатор правила.
    - `name`: задайте название правила.
    - `version`: укажите версию правила, следуя стандартам Semantic Versioning.
    - `type`: укажите тип правила как `normalization_rule`.
    - `author` (опционально): укажите имя и контактные данные автора.
    - `description` (опционально): предоставьте краткое описание назначения и функциональности

правила.

- `data_source` (опционально): укажите список совместимых с правилом источников данных для классификации правила в системе.
- `tags` (опционально): добавьте теги для классификации и быстрого поиска правила.

2. Разработайте логику фильтрации событий:
   - `filter`: создайте условие для фильтрации входящих событий. События, которые не удовлетворяют условиям фильтра, исключаются из дальнейшего процесса нормализации.
3. Определите правила нормализации:
   - `normalizer`: создайте блок правил на языке VRL, который будет применяться для нормализации выбранных событий.
4. Разработайте тесты для проверки правила (опционально):
   - `tests`: создайте тесты, включающие различные типы событий и условия, для проверки корректности работы правила нормализации.

Пример правила нормализации

Пример 75. Правило нормализации, схема RObject

```
# Уникальный идентификатор правила.
id: example/basic_normalization_rule

# Название правила.
name: BasicNormalizationRule

# Имя автора правила.
author: John Doe <johndoe@example.com>

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Базовое правило нормализации

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - logs/default.log
    - logs/somelog.log

# Теги, связанные с правилом.
tags:
  - normalization
  - security

# Тип элемента экспертизы.
```

```
type: normalization_rule

# Блок правил на языке VRL с условиями фильтрации событий.
filter: !vrl |
  .event.host == "170.174.42.191"

# Блок с правилами на языке VRL для нормализации полей события.
normalizer: !vrl |
    e = parse_json(.event) ?? {}
    . |= {
      "userId": e."user-identifier",
      "timestamp": parse_timestamp(e.datetime, "%d/%b/%Y:%H:%M:%S") ?? now(),
      "host": e.host,
      "method": e.method,
      "path": e.request,
      "protocol": e.protocol
    }

    if e.status == "404" || e.status == "401" {
      .responseStatus = "Bad request"
    }

    if e.protocol == "HTTP/2.0" {
        .protocol = "HTTP/2"
    }
```

Интерпретация

Поле `filter`:

- `.event.host == "198.51.100.191"`:

  Сравнения поля `host` со значением "198.51.100.191" в сыром событии. Если значение поля `host` в сыром событии совпадает с "198.51.100.191", то событие будет нормализовано согласно правилам в поле `normalizer`. Если значение поля `host` не совпадает с указанным IP-адресом, правила нормализации к событию применены не будут.

Поле `normalizer`:

- `e = parse_json(.event) ?? {}`:

  Попытка преобразовать содержимое поля `.event` из формата JSON в структуру данных VRL.

  Если преобразование не удалось или результатом является `null`, переменной `e` будет присвоен пустой объект `{}`, что достигается с помощью оператора `??`.

- `. |= { ⋯ }`:

  Оператор обновления/добавления новых полей к текущему объекту на основе значений из объекта `e`.

Структура, которая присваивается текущему объекту:

- `userId`: присваивает значение поля `user-identifier` из объекта `e` к полю `userId` текущего объекта.
- `timestamp`: преобразует строковое значение поля `datetime` из объекта `e` в метку времени с использованием указанного формата. Если преобразование не удастся или значение поля `datetime` равно `null`, будет использовано текущее время, возвращаемое функцией `now()`.
- `host`: присваивает значение поля `host` из объекта `e` к полю `host` текущего объекта.
- `method`: присваивает значение поля `method` из объекта `e` к полю `method` текущего объекта.
- `path`: присваивает значение поля `request` из объекта `e` к полю `path` текущего объекта.
- `protocol`: присваивает значение поля `protocol` из объекта `e` к полю `protocol` текущего объекта.

`if e.status == "404" || e.status == "401"`:

- Условное выражение проверяет, равно ли значение поля `status` из объекта `e` строке "404" или "401".

`.responseStatus = "Bad request"`:

- Если условие выше выполняется, то поле `responseStatus` текущего объекта устанавливается в значение "Bad request".

`if e.protocol == "HTTP/2.0"`:

- Условное выражение проверяет, равно ли значение поля `protocol` из объекта `e` строке "HTTP/2.0".

`.protocol = "HTTP/2"`:

- Если условие выше выполняется, то поле `protocol` текущего объекта заменяется на значение "HTTP/2".

Обработка событий правилом нормализации "BasicNormalizationRule"

Рассмотрим работу правила на примере разбора базового (сырого) события.

Допустим, на нормализатор поступают события следующего вида:

Пример 76. Исходное событие, JSON

```
{
  "event": "{\"host\":\"198.51.100.191\",\"user-
identifier\":\"JohnDoe\",\"datetime\":\"31/Jul/2023:06:55:49\",\"method\":\"GET\",\"re
quest\":\"/observability/metrics/production\",\"protocol\":\"HTTP/2.0\",\"status\":\"4
04\",\"bytes\":22427,\"referer\":\"https://example.url.com\"}"
}
```

После применения к событию правила "BasicNormalizationRule", оно принимает следующий вид:

Пример 77. Нормализованное событие, JSON

```
{
  "event": "{\"host\":\"198.51.100.191\",\"user-
identifier\":\"JohnDoe\",\"datetime\":\"31/Jul/2023:06:55:49\",\"method\":\"GET\",\"re
```

```
quest\":\"/observability/metrics/production\",\"protocol\":\"HTTP/2.0\",\"status\":\"4
04\",\"bytes\":22427,\"referer\":\"https://example.url.com\"}",
  "host": "170.174.42.191",
  "method": "GET",
  "path": "/observability/metrics/production",
  "protocol": "HTTP/2",
  "referer": "https://example.url.com",
  "responseBytes": 22427,
  "responseStatus": "Bad request",
  "serverIP": "198.51.100.191",
  "timestamp": "2023-07-30T23:55:49Z",
  "userAgent": null,
  "userId": "JohnDoe"
}
```

Событие удовлетворяет условию из поля `filter`, следовательно, оно подлежит нормализации и обрабатывается согласно правилам из поля `normalizer`. В структуре нормализованного события в поле `event` записывается исходное событие. Прочие поля добавляются к объекту и определяются в соответствии с правилами нормализации, хранящимся в поле `normalizer`.

Расширение конфигурации существующего правила нормализации

Рассмотрим сценарий, в котором требуется расширить конфигурацию существующего правила "BasicNormalizationRule" таким образом, чтобы помимо базового преобразования полей времени, источника и статуса события, оно также устанавливало уровень угрозы в отдельном поле в зависимости от значения в поле `status`. Также добавим условия для разбора остальных полей события, которые не обрабатываются в исходном правиле нормализации.

Допустим, исходное правило нормализации "BasicNormalizationRule" выглядит следующим образом:

▼ *Исходное правило нормализации BasicNormalizationRule, схема RObject*

```
# Уникальный идентификатор правила.
id: example/basic_normalization_rule

# Название правила.
name: BasicNormalizationRule

# Имя автора правила.
author: John Doe <johndoe@example.com>

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Базовое правило нормализации

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
```

```
  - source: Sysmon
  - events:
    - logs/default.log
    - logs/somelog.log

# Теги, связанные с правилом.
tags:
  - normalization
  - security

# Тип элемента экспертизы.
type: normalization_rule

# Блок правил на языке VRL с условиями фильтрации событий.
filter: !vrl |
  .event.host == "170.174.42.191"

# Блок с правилами на языке VRL для нормализации полей события.
normalizer: !vrl |
    e = parse_json(.event) ?? {}
    . |= {
      "userId": e."user-identifier",
      "timestamp": parse_timestamp(e.datetime, "%d/%b/%Y:%H:%M:%S") ?? now(),
      "host": e.host,
      "method": e.method,
      "path": e.request,
      "protocol": e.protocol
    }

    if e.status == "404" || e.status == "401" {
      .responseStatus = "Bad request"
    }

    if e.protocol == "HTTP/2.0" {
        .protocol = "HTTP/2"
    }
```

Тогда расширенная конфигурация правила на основе правила будет выглядеть следующим образом:

Пример 78. Конфигурация расширенного правила, схема RObject

```
# Уникальный идентификатор правила.
id: example/enhanced_normalization_rule

# Название правила.
name: EnhancedNormalizationRule

# Имя автора правила.
author: John Doe <johndoe@example.com>

# Версия правила.
```

```
version: 1.0.0

# Описание назначения правила.
description: Улучшенное правило нормализации JSON-события с определением уровня угрозы
в поле `severity`.

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
     - logs/default.log
     - logs/somelog.log

# Теги, связанные с правилом.
tags:
  - security
  - normalization

# Тип элемента экспертизы.
type: normalization_rule

# Блок правил на языке VRL с условиями фильтрации событий.
filter: !vrl |
  .event.host == "170.174.42.191"

# Блок с правилами на языке VRL для нормализации полей события.
normalizer: !vrl |
  e = parse_json(.event) ?? {}
  . |= {
    "userId": e."user-identifier",
    "timestamp": parse_timestamp(e.datetime, "%d/%b/%Y:%H:%M:%S") ?? now(),
    "host": e.host,
    "method": e.method,
    "path": e.request,
    "protocol": e.protocol,
    "responseStatus": e.status,
    "responseBytes": e.bytes,
    "referer": e.referer,
    "userAgent": e."user-agent",
    "serverIP": "170.174.42.191",
    "severity": e.severity,
  }

  if e.status == "404" || e.status == "401" {
    .responseStatus = "Bad request"
  }

  if e.status == "404" || e.status == "401" {
    .severity = "High"
```

```
  }
```

Улучшенное правило "EnhancedNormalizationRule" представляет собой расширенную версию исходного правила. Оно разбирает недостающие поля сырого события и добавляет поле `severity`, которое определяет уровень угрозы в зависимости от статуса события. Если статус события соответствует "404" или "401", в поле `severity` устанавливается значение "High". В противном случае полю `severity` будет присвоено "null".

Таким образом, после применения правила "EnhancedNormalizationRule", нормализованное событие принимает следующий вид:

Пример 79. Событие, обработанное правилом "EnhancedNormalizationRule", JSON

```
= {
  "event": "{\"host\":\"170.174.42.191\",\"user-
identifier\":\"JohnDoe\",\"datetime\":\"31/Jul/2023:06:55:49\",\"method\":\"GET\",\"re
quest\":\"/observability/metrics/production\",\"protocol\":\"HTTP/2.0\",\"status\":\"4
04\",\"bytes\":22427,\"referer\":\"https://example.url.com\"}",
  "host": "170.174.42.191",
  "method": "GET",
  "path": "/observability/metrics/production",
  "protocol": "HTTP/2.0",
  "referer": "https://example.url.com",
  "responseBytes": 22427,
  "responseStatus": "Bad request",
  "serverIP": "170.174.42.191",
  "severity": "High", # Добавлено поле severity, уровень угрозы установлен в значение
High.
  "timestamp": "2023-07-31T06:55:49Z",
  "userAgent": null,
  "userId": "JohnDoe"
}
```

Использование глобальных функций

Если одинаковые преобразования данных нужно выполнять в нескольких правилах, вы можете создать глобальную функцию с необходимой логикой. Создадим для указанного примера глобальную функцию со следующим исполняемым кодом и аргументами:

```
...
# Имя функции.
name: set_severity
...

# Основной исполняемый код.
source: !vrl |
  if status == "404" || status == "401" {
    "High"
  } else {
```

```
    null
  }

# Аргументы функции.
arguments:
  - keyword: status
    kind: bytes
...
```

Тогда в правиле нормализации можно вызывать глобальную функцию так:

```
...
.severity = set_severity(e.status)
```

Правила агрегации

В данном разделе представлен обзор структуры, принципов работы, процесса создания и расширения конфигурации правил агрегации в системе. Управление правилами агрегации в системе осуществляется в разделе **Экспертиза**.

О правилах агрегации

Правило агрегации — это структурированный объект данных в формате RObject (.ro), который описывает логику объединения и анализа групп событий безопасности.

Структура правила агрегации включает определение условий фильтрации событий, параметров группировки, условий начала и завершения агрегации, а также стратегий агрегации для каждого поля. Фильтрация событий осуществляется на основе заранее заданных критериев, определяющих, какие события будут включены в процесс агрегации. Условия начала и завершения агрегации определяют логику работы агрегационного окна, определяя, когда и как будет открываться и закрываться окно.

Важной частью правила является определение группировки событий, что позволяет собирать события в логически согласованные группы для последующей агрегации. Группировка может осуществляться по различным полям события, например, содержащим данные об IP-адресах, идентификаторах устройств или пользователях. Если указано несколько полей для группировки, то события будут группироваться по уникальным комбинациям значений этих полей.

Стратегии агрегации для каждого поля определяют, как будут объединяться значения соответствующих полей событий в процессе агрегации. Доступны различные стратегии, такие как суммирование, выбор максимального и минимального значений, различные формы конкатенации строк, такие как обычное соединение, соединение с новой строки или без разделителей, а также методы для работы с массивами, включая добавление всех значений, формирование массива уникальных значений и выбор самого длинного или короткого массива. Кроме того, существуют стратегии для сохранения значений из первого или последнего события в агрегации.

Работа с правилом агрегации

Доступные операции над правилом агрегации:

- Создание правила агрегации
- Изменение правила агрегации
- Удаление правила агрегации

Создание правила агрегации

Чтобы добавить правило агрегации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Правило агрегации**. Система отобразит окно создания правила агрегации.

При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля правила агрегации, чтобы определить логику его работы.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст правило агрегации и отобразит соответствующее уведомление. Новое правило появится в таблице раздела **Экспертиза**.

Чтобы правило агрегации стало доступно на конвейерах событий, его необходимо включить.

Вы можете создать черновик правила агрегации без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик правила и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Изменение правила агрегации

Чтобы изменить правило агрегации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку правила агрегации в списке. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек правила.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

4. Внесите требуемые изменения в конфигурацию правила агрегации.
5. Нажмите на кнопку **Обновить версию**. Система опубликует новую версию правила агрегации с измененной конфигурацией и отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию правила агрегации в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки правила будет отображаться предупреждение о наличии неопубликованных изменений.

Удаление правила агрегации

- Удаление доступно только для элементов экспертизы с типом создания **Пользовательский**.
- Если элемент экспертизы используется в других сущностях системы, его удаление невозможно.

Чтобы удалить правило агрегации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку правила агрегации в списке. Система отобразит в правой части экрана карточку этого правила с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления правила.
4. Нажмите на кнопку **Удалить**. Система удалит правило агрегации и отобразит соответствующее уведомление. Удаленное правило будет исключено из списка элементов экспертизы.

Вы также можете удалить группу элементов экспертизы. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные элементы и отобразит соответствующее уведомление. Удаленные элементы будет исключены из списка элементов экспертизы.

Структура правила агрегации

Правило агрегации в системе представляет собой структурированный объект данных в формате RObject (.ro), который включает в себя ряд обязательных и опциональных полей для определения процесса агрегации.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор правила агрегации.	строка	да
`name`	Название правила агрегации.	строка	да
`type`	Тип элемента. Всегда имеет значение `aggregation_rule` для элементов экспертизы этого типа.	строка	да
`version`	Версия правила агрегации, представленная в формате Semantic Versioning.	строка	да
`author`	Имя и контактные данные автора правила агрегации.	строка	нет
`description`	Описание правила агрегации, содержащее дополнительную информацию о его назначении и функциональности.	строка	нет
`data_source`	Список источников данных, с которыми совместимо правило агрегации. Используется для классификации и быстрого поиска правил в системе.	массив строк массив объектов	нет
`tags`	Список тегов, присвоенных правилу агрегации для классификации и быстрого поиска.	массив строк	нет
`filter`	Фильтр на языке VRL, определяющий условия, при которых события будут агрегироваться.	строка	да
`group_by`	Поля, по которым будут группироваться агрегированные события. Если ничего не указано — все поступающие события будут объединяться в одну группу.	массив строк	нет
`starts_when`	Условие на языке VRL, при выполнении которого открывается новое агрегационное окно. Нельзя использовать одновременно с параметром `ends_when`.	строка	нет
`max_events`	Максимальное количество событий для агрегации.	число	нет
`expire_after_ms`	Время в миллисекундах, после которого агрегационное окно закрывается, если не поступают новые события.	число	нет

Поле	Описание	Тип данных	Обязательное поле
`flush_period_ms`	Интервал в миллисекундах для проверки и закрытия агрегационных окон.	число	нет
`merge_strategies`	Стратегии, определяющие, как будут объединяться значения полей из агрегированных событий.	объект	нет
`ends_when`	Условие на языке VRL, при выполнении которого агрегационное окно закрывается. Нельзя использовать одновременно с параметром `starts_when`.	строка	нет
`tests`	Блок тестирования правила агрегации, содержащий массив тестовых событий и условия для проверки срабатывания правила.	объект	нет

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform> # Платформа или информационная система.
- <source> # Источник событий.
# События (EventId) или путь к журналу событий:
- <EventID_1>
- <EventID_2>
```

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `tests`

Поле	Описание	Тип данных	Обязательное поле
`name`	Название теста, позволяющее идентифицировать его в тестовой выдаче.	строка	да
`events`	Поток тестовых событий, обрабатываемых правилом.	массив	да

Поле	Описание	Тип данных	Обязательное поле
`assertion`	Блок правил на языке VRL, который будет вызываться на каждое событие, испускаемое полем `events`. Используя встроенные функции `assert` и `assert_eq`, можно строить проверки на соответствие испускаемого события тестовым условиям.	строка	да

Опции `merge_strategies`

Значение, указанное в `merge_strategies`, определяет метод, с помощью которого значения из полей разных событий будут объединяться в ходе агрегации.

Опция	Описание
`array`	Создает массив, включающий все встречающиеся значения поля из агрегированных событий. Используется, когда необходимо сохранить все значения без исключения.
`concat`	Объединяет строковые значения в одну строку с разделением пробелами. Подходит для ситуаций, где необходимо сформировать единое текстовое представление из разных строковых значений.
`concat_newline`	Аналогично `concat`, но использует перенос строки в качестве разделителя. Это облегчает различение отдельных значений в одной длинной строке.
`concat_raw`	Соединяет строковые значения без какого-либо разделителя между ними. Эта стратегия полезна, когда необходимо сформировать непрерывную строку данных.
`discard`	Игнорирует все значения, кроме первого встретившегося. Полезно, когда важно сохранить только первоначальное значение поля из всей группы событий.
`flat_unique`	Формирует массив уникальных значений, исключая повторения. Используется для идентификации уникальных характеристик в группе событий.
`longest_array`	Выбирает массив максимальной длины из всех массивов, встречающихся в агрегированных событиях. Применяется для определения наиболее обширного набора данных.
`max`	Находит и сохраняет максимальное числовое значение среди всех агрегированных событий. Это может быть полезно для определения пиковых значений.
`min`	Аналогично `max`, но для определения минимального числового значения.
`retain`	Сохраняет значение из последнего события в агрегированной группе. Это полезно для отслеживания самых последних изменений.

Опция	Описание
`shortest_array`	Выбирает массив минимальной длины, что может быть полезно для определения наименее обширного набора данных в группе.
`sum`	Вычисляет сумму всех числовых значений поля. Эта стратегия широко используется для агрегации количественных данных, таких как счетчики или объемы.

Составление правила

Для составления правила агрегации в системе, необходимо определить ключевые элементы структуры правила. Эти элементы включают поля метаданных и поля, управляющие логикой работы правила.

1. Определите метаданные правила:
    - `id`: задайте уникальный идентификатор правила.
    - `name`: задайте название правила.
    - `version`: укажите версию правила, следуя стандартам Semantic Versioning.
    - `type`: определите тип правила как `aggregation_rule`.
    - `author` (опциональное): укажите имя и контактные данные автора.
    - `description`(опциональное): предоставьте краткое описание назначения и функциональности правила.
    - `data_source` (опциональное): укажите список совместимых с правилом источников данных для классификации и фильтрации.
    - `tags` (опциональное): добавьте теги для классификации и поиска правила.
2. Разработайте логику работы правила:
    - `filter` (обязательное): создайте условие для фильтрации событий.
    - `group_by` (обязательное): определите поля для группировки событий.
    - `starts_when` и `ends_when` (опциональные): настройте условия открытия и закрытия окна агрегации. В рамках правила допускается одновременное использование только одного из параметров.
    - `expire_after_ms` и `flush_period_ms` (опциональные): установите временные параметры для управления агрегационными окнами.
    - `max_events` (опциональное): укажите максимальное количество событий для агрегации.
    - `merge_strategies` (опциональное): выберите стратегии агрегации для полей событий.
3. Разработайте тесты для правила (опциональное):
    - `tests`: создайте тесты, включая сценарии и условия, для проверки работы правила.

Поля метаданных

Метаданные являются основой для идентификации и описания правила агрегации в системе. Они должны быть заполнены следующим образом:

- `id`: установите уникальный идентификатор правила. Этот идентификатор должен быть уникальным в рамках системы.
- `name`: укажите название правила.
- `version`: задайте версию правила, следуя принципам Semantic Versioning (например, 1.0.0).
- `type`: установите тип правила `aggregation_rule`.
- `author` (опциональное): укажите имя и контактные данные создателя правила.
- `description` (опциональное): предоставьте краткое описание назначения и функциональности правила.
- `data_source` (опциональное): укажите список совместимых с правилом источников данных.
- `tags` (опциональное): добавьте теги для классификации правила в системе.

Поле `filter`

Поле `filter` служит для определения критериев отбора событий, которые будут включены в процесс агрегации. Фильтрация основана на языке VRL, предоставляя гибкие возможности для анализа и обработки данных событий.

Пример:

```
filter: !vrl |
  .event_type == "login_failure" && .source_ip != "192.168.1.1"
```

В этом примере выбираются события с типом "login_failure", исключая события от конкретного IP-адреса.

Поле `group_by`

Поле `group_by` задает основу для группировки событий в рамках агрегации. Можно указать одно или несколько полей, по которым события будут группироваться, что позволяет более точно анализировать сходные или связанные данные. Если ничего не указано — все поступающие события будут объединяться в одну группу.

Пример:

```
group_by:
  - source_ip
  - user_agent
```

Здесь события группируются по IP-адресу источника и типу пользовательского агента.

Поле `starts_when`

`starts_when` определяет условие, при котором откроется новое окно агрегации. Это условие задает начальную точку для создания новой группы событий.

Параметр `starts_when` нельзя использовать одновременно с параметром `ends_when`

в рамках одного правила агрегации.

Пример:

```
starts_when: !vrl |
  .event_type == "network_connection" && .port == 443
```

Агрегация начнется, когда обнаружится событие типа "network_connection" на порту 443.

Поле `ends_when`

`ends_when` определяет условие для закрытия окна агрегации. Это условие может быть основано на различных критериях, соответствующих потребностям безопасности и анализа.

Параметр `ends_when` нельзя использовать одновременно с параметром `starts_when` в рамках одного правила агрегации.

Пример:

```
ends_when: !vrl |
  .event_type == "network_disconnection"
```

Здесь агрегация завершается, когда появляется событие разъединения сети.

Поле `expire_after_ms`

`expire_after_ms` устанавливает максимальный период ожидания для агрегационного окна в миллисекундах. Это поле помогает контролировать продолжительность агрегации и гарантирует, что данные обрабатываются в разумные временные рамки.

Пример:

```
expire_after_ms: 30000
```

Агрегационное окно будет закрыто, если новые события не появятся в течение 30 секунд.

Поле `flush_period_ms`

`flush_period_ms` задает периодичность проверки и закрытия агрегационных окон. Этот параметр важен для обеспечения своевременной обработки данных и управления памятью.

Пример:

```
flush_period_ms: 10000
```

Система будет проверять и закрывать агрегационные окна каждые 10 секунд.

Поле `max_events`

`max_events` определяет максимальное количество событий, которое может быть включено в одно агрегационное окно. Этот параметр помогает контролировать объем данных в каждой агрегации.

Пример:

```
max_events: 20
```

Агрегационное окно будет закрыто после обработки 20 событий.

Поле `merge_strategies`

`merge_strategies` определяет стратегии обработки и объединения данных из различных событий в процессе агрегации. Эти стратегии позволяют гибко управлять данными, создавая информативные и ценные агрегированные записи.

Если для указанного поля не определена стратегия, будет использоваться поведение по умолчанию.

Поведение по умолчанию:

- Первое значение строкового поля сохраняется, а последующие значения отбрасываются.
- Для полей временных меток сохраняется первое значение, и добавляется новое поле `[имя-поля]_end` с временной меткой последнего полученного значения.
- Числовые значения суммируются.

Пример:

```
merge_strategies:
  bytes_sent: sum
  messages: concat_newline
```

Здесь значения поля `bytes_sent` суммируются, а строки в поле `messages` объединяются с разделителем в виде новой строки.

Поле `tests`

Поле `tests` включает набор тестов для проверки и верификации правила агрегации. Эти тесты помогают убедиться, что правило работает корректно и соответствует ожидаемым результатам.

Определите параметры теста:

- В `name` укажите название теста.
- В `events` создайте массив событий, которые будут использоваться для тестирования. Эти события должны имитировать поток реальных исходных событий, которые поступают на агрегатор.
- В `assertion` устанавливается условие или набор условий, которые должны быть выполнены в результате обработки тестовых событий правилом. Это может включать проверку конкретных

полей в итоговых агрегированных событиях или подсчет количества таких событий.

Тестовые события

Тестовые события должны имитировать данные, которые в реальности поступают на агрегатор. Они должны соответствовать формату и структуре событий, обрабатываемых правилом агрегации и содержать поля, которые используются в правиле.

Условия проверки

Условия должны точно отражать ожидаемый результат применения правила к тестовым событиям. Это может быть проверка наличия определенных полей в событии, соответствие их значений ожидаемым или сравнение количества событий с предполагаемым исходом. Для более сложных правил могут потребоваться более сложные утверждения, которые могут включать логические операции и сравнения.

Пример:

```
tests:
  - name: AggregationTest
    events:
      - { "event_type": "login_failure", "source_ip": "192.168.1.1" }
      - { "event_type": "login_failure", "source_ip": "192.168.1.2" }
    assertion: !vrl |
      assert_eq!(.aggregated_event_count, 2)
```

Здесь проверяется, что правило агрегирует два события типа "login_failure".

Структура агрегированного события

В результате выполнения агрегации создается агрегированное событие. Помимо полей, которые явно определяются правилом агрегации, структура агрегированного события содержит ряд служебных полей, заполняемых по умолчанию.

Ключ поля	Тип данных	Описание
`id`	UUID	ID агрегированного события. Генерируется автоматически при создании события.
`sourceIp`	IPv6	IP-адрес источника базового события. По умолчанию записывается значение для первого базового события в агрегации.
`tenantId`	LCString	ID тенанта, к которому относится агрегированное событие.
`art`	DateTime	Дата и время получения базового события. По умолчанию записывается значение для первого базового события в агрегации.
`raw`	String	Текст базового события. По умолчанию записывается значение для первого базового события в агрегации.

Ключ поля	Тип данных	Описание
collectorId	LCString	ID коллектора, на котором создается агрегированное событие.
timestamp	DateTime	Дата и время создания агрегированного события.
aggregationRuleName	LCString	Название правила агрегации, обработавшего базовое событие.
aggregationRuleId	LCString	ID правила агрегации, обработавшего базовое событие.
type	Enum	Тип события. По умолчанию устанавливается значение "2", соответствующее типу "агрегированное событие".
cnt	UInt32	Количество базовых событий, на основе которых создается агрегированное событие.
baseEventIds	Массив UUID	Список ID базовых событий, на основе которых создается агрегированное событие.
baseEventTimestamps	Массив DateTime	Список дат и времен получения базовых событий, на основе которых создается агрегированное событие.
groupedBy	Массив LCString	Список полей модели событий, по которому группируются базовые события при агрегации.

Поля id, tenantId, collectorId, timestamp, type, cnt, baseEventIds, baseEventTimestamps заполняются системой автоматически. Их правила заполнения не могут быть изменены.

Поля groupedBy, aggregationRuleId и aggregationRuleName заполняются на основе настроек, заданных в правиле агрегации.

При необходимости вы можете изменить правила заполнения остальных полей. Для этого задайте соответствующие правила в блоке merge_strategies правила агрегации.

Прочие служебные и регулярные поля модели события заполняются в соответствии с настройками, заданными в блоке merge_strategies правила агрегации.

Пример правила агрегации

Пример 80. Правило агрегации, схема RObject

```
# Уникальный идентификатор правила.
id: example/network_activity_rule

# Название правила.
name: NetworkActivityAggregationRule

# Тип элемента экспертизы.
type: aggregation_rule

# Версия правила.
```

```
version: 1.0.0

# Описание назначения правила.
description: Агрегация событий сетевой активности для анализа трафика

# Имя и контактные данные автора правила.
author: John Doe <johndoe@example.com>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_1
    - EventID_2

# Теги, связанные с правилом.
tags:
  - network
  - traffic_analysis

# Отбрасывает события, которые не соответствуют критериям.
filter: !vrl |
  # Фильтрует события, не относящиеся к категории "Network" или имеющие низкий уровень
серьезности.
  .category == "Network" &&
  to_int!(.severity) > 3

# Группировка событий по источнику и назначению.
group_by:
  - src_ip
  - dest_ip

# Условие открытия агрегационного окна.
start_when: !vrl |
  # Начинаем агрегацию при обнаружении события начала соединения.
  .event_type == "connection_start"

# Максимальное количество событий.
max_events: 5

# Время ожидания последнего события (миллисекунды).
expire_after_ms: 60000

# Интервал проверки окон агрегации (миллисекунды).
flush_period_ms: 2000

# Стратегии агрегации значений для каждого указанного поля.
merge_strategies:
  bytes_sent: sum
  bytes_received: sum
```

```
  severity: max

# Тесты для проверки работы правила.
tests:
 - name: Network Activity Test
   events:
   - {"category":"Network", "severity":5, "event_type":"connection_start",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":1500,
"bytes_received":2000}
   - {"category":"Network", "severity":4, "event_type":"data_transfer",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":3000,
"bytes_received":4000}
   - {"category":"Network", "severity":5, "event_type":"connection_end",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":500, "bytes_received":600}
   assertion: !vrl |
     # Проверяем, что агрегированные значения соответствуют ожиданиям.
     assert_eq!(.[0], {
       "category": "Network",
       "event_type": "connection_start",
       "src_ip": "192.0.2.10",
       "dest_ip": "192.0.2.20",
       "bytes_sent": 5000,
       "bytes_received": 6600,
       "severity": 5
     })
```

Интерпретация

Правило агрегации "NetworkActivityAggregationRule" служит для агрегации сетевых событий, соответствующих передаче данных в рамках одного сетевого соединения.

Метаданные правила:

- `id`: уникальный идентификатор правила в системе.
- `name`: название правила.
- `type`: тип `aggregation_rule` указывает на то, что это правило агрегации.
- `version`: версия правила.
- `description`: описывает цель правила — "Агрегация событий сетевой активности для анализа трафика".
- `author`: создатель правила.
- `tags`: теги `network` и `traffic_analysis` классифицируют правило по его функциональности и области применения.

Фильтрация событий (filter):

- `filter`: фильтрует события по категории "Network" и уровню серьезности более 3. Это сужает область применения правила только к событиям, которые удовлетворяют условиям фильтрации.

Группировка событий (group_by):

- `group_by`: группирует события по полям `src_ip` и `dest_ip`. Это позволяет анализировать взаимодействия между конкретными источниками и назначениями в сети.

Условие начала агрегации (starts_when):

- `starts_when`: определяет начало агрегации при появлении событий типа "connection_start", что является признаком начала нового сетевого соединения.

Параметры агрегационного окна (max_events, expire_after_ms, flush_period_ms):

- `max_events`: ограничивает количество событий в агрегации до 5. После достижения этого количества, агрегационное окно закрывается и генерируется агрегированное событие.
- `expire_after_ms`: устанавливает максимальный период ожидания последнего события в агрегационном окне в 60000 миллисекунд (1 минута).
- `flush_period_ms`: задает частоту проверки и закрытия агрегационных окон в 2000 миллисекунд (2 секунды).

Стратегии агрегации (merge_strategies):

- `merge_strategies`: определяет, как будут объединяться поля событий. Для "bytes_sent" и "bytes_received" применяется стратегия суммирования (`sum`).

Тестирование правила (tests):

- `name`: "Network Activity Test" — название теста.
- `events`: описывает ряд тестовых событий, имитирующих реальный сетевой трафик, для проверки эффективности правила.
- `assertion`: проверяет, что правило корректно агрегирует события и вычисляет суммы переданных и полученных байтов. Это подтверждает, что правило работает как предполагалось и агрегирует данные согласно заданным критериям.

Обработка событий правилом агрегации "NetworkActivityAggregationRule"

Проанализируем работу правила агрегации "NetworkActivityAggregationRule", фокусируясь на процессе обработки событий.

Представим, что на агрегатор поступает поток сетевых событий, которые соответствуют критериям, заданным в фильтре `filter` правила. Эти события содержат данные о типе сетевого соединения, IP-адресах источника и назначения, а также объеме переданных и полученных данных.

```
{
  "category": "Network",
  "severity": 5,
  "event_type": "connection_start",
  "src_ip": "192.168.1.10",
  "dest_ip": "192.168.1.20",
  "bytes_sent": 1500,
  "bytes_received": 2000
```

```
}
```

По мере поступления этих событий, они фильтруются правилом "NetworkActivityAggregationRule". Правило агрегирует события, группируя их по IP-адресам источника и назначения (`group_by`). Таким образом, события, исходящие от одной пары IP-адресов, объединяются в одно агрегированное событие.

Агрегация инициируется, когда обнаруживается событие типа `connection_start` (`starts_when`). Правило продолжает агрегировать события до тех пор, пока не будет достигнуто максимальное количество событий (`max_events`) или не истечет время ожидания (`expire_after_ms`).

Агрегированное событие включает в себя сумму значений `bytes_sent` и `bytes_received` для всех событий в группе (`merge_strategies`).

```
{
  "event_type": "aggregated_network_activity",
  "src_ip": "192.168.1.10",
  "dest_ip": "192.168.1.20",
  "total_bytes_sent": 5000,
  "total_bytes_received": 6600
}
```

Данное агрегированное событие отражает общую активность между двумя конкретными IP-адресами, позволяя идентифицировать аномалии или необычную активность в сети.

Расширение конфигурации существующего правила

Рассмотрим сценарий, в котором необходимо расширить функциональность правила агрегации "NetworkActivityAggregationRule", чтобы улучшить анализ сетевого трафика и повысить точность агрегации событий.

▼ *Исходное правило агрегации "NetworkActivityAggregationRule", схема RObject*

```
# Уникальный идентификатор правила.
id: example/network_activity_rule

# Название правила.
name: NetworkActivityAggregationRule

# Тип элемента экспертизы.
type: aggregation_rule

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Агрегация событий сетевой активности для анализа трафика

# Имя и контактные данные автора правила.
```

```
author: John Doe <johndoe@example.com>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_1
    - EventID_2

# Теги, связанные с правилом.
tags:
  - network
  - traffic_analysis

# Отбрасывает события, которые не соответствуют критериям.
filter: !vrl |
  # Фильтрует события, не относящиеся к категории "Network" или имеющие низкий
уровень серьезности.
  .category == "Network" &&
  to_int!(.severity) > 3

# Группировка событий по источнику и назначению.
group_by:
  - src_ip
  - dest_ip

# Условие открытия агрегационного окна.
start_when: !vrl |
  # Начинаем агрегацию при обнаружении события начала соединения.
  .event_type == "connection_start"

# Максимальное количество событий.
max_events: 5

# Время ожидания последнего события (миллисекунды).
expire_after_ms: 60000

# Интервал проверки окон агрегации (миллисекунды).
flush_period_ms: 2000

# Стратегии агрегации значений для каждого указанного поля.
merge_strategies:
  bytes_sent: sum
  bytes_received: sum
  severity: max

# Тесты для проверки работы правила.
tests:
 - name: Network Activity Test
   events:
```

```
    - {"category":"Network", "severity":5, "event_type":"connection_start",
  "src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":1500,
  "bytes_received":2000}
    - {"category":"Network", "severity":4, "event_type":"data_transfer",
  "src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":3000,
  "bytes_received":4000}
    - {"category":"Network", "severity":5, "event_type":"connection_end",
  "src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "bytes_sent":500,
  "bytes_received":600}
    assertion: !vrl |
      # Проверяем, что агрегированные значения соответствуют ожиданиям.
      assert_eq!(.[0], {
        "category": "Network",
        "event_type": "connection_start",
        "src_ip": "192.0.2.10",
        "dest_ip": "192.0.2.20",
        "bytes_sent": 5000,
        "bytes_received": 6600,
        "severity": 5
      })
```

Для этого добавим новые условия группировки и стратегии агрегации, а также дополнительные параметры для более тонкой настройки агрегации событий.

Пример 81. Пример расширенного правила "EnhancedNetworkActivityAggregationRule", схема RObject

```
# Уникальный идентификатор правила.
id: example/network_activity_rule_enhanced

# Название правила.
name: EnhancedNetworkActivityAggregationRule

# Тип элемента экспертизы.
type: aggregation_rule

# Версия правила.
version: 1.0.0

# Описание назначения правила.
description: Расширенная агрегация событий сетевой активности для детального анализа
трафика

# Имя и контактные данные автора правила.
author: John Doe <johndoe@example.com>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_1
```

```
    - EventID_2

# Теги, связанные с правилом.
tags:
  - network
  - advanced_traffic_analysis

#  Условия для фильтрации событий.
filter: !vrl |
  # Фильтрация событий по категории "Network" и высокому уровню серьезности, а также
специфическим характеристикам запросов.
  .category == "Network" && .severity > 3 && match(.user_agent, "^Mozilla")

# Расширенные параметры группировки событий.
group_by:
  - src_ip
  - dest_ip
  - user_agent
  - http_method

# Условие открытия агрегационного окна.
start_when: !vrl |
  # Агрегация начинается при обнаружении начала соединения или определенных типов
HTTP-запросов.
  .event_type == "connection_start" || includes(["POST", "GET"], .http_method)

# Увеличенное максимальное количество событий.
max_events: 10

# Измененное время ожидания последнего события (миллисекунды).
expire_after_ms: 120000

# Интервал проверки окон агрегации (миллисекунды).
flush_period_ms: 3000

# Расширенные стратегии агрегации значений.
merge_strategies:
  bytes_sent: sum
  bytes_received: sum
  request_paths: longest_array

# Тесты для проверки работы расширенного правила.
tests:
 - name: Enhanced Network Activity Test
   events:
   - {"category":"Network", "severity":5, "event_type":"connection_start",
"src_ip":"192.0.2.10", "dest_ip":"192.0.2.20", "user_agent": "Mozilla/5.0",
"http_method": "POST", "bytes_sent":2000, "bytes_received":3000}
   - {"category":"Network", "severity":4, "event_type":"data_transfer",
"src_ip":"192.0.2.10", "dest_ip":192.0.2.20", "user_agent": "Mozilla/5.0",
"http_method": "GET", "bytes_sent":4000, "bytes_received":5000}
```

```
  assertion: !vrl |
    # Проверяем, что агрегированные значения соответствуют ожиданиям.
    assert_eq!(.[0], {
      "category": "Network",
      "event_type": "connection_start",
      "src_ip": "192.0.2.10",
      "dest_ip": "192.0.2.20",
      "user_agent": "Mozilla/5.0",
      "http_method": "POST",
      "bytes_sent": 6000,
      "bytes_received": 8000,
      "request_paths": "/api/load"
    })
```

Изменения, внесенные в улучшенное правило "EnhancedNetworkActivityAggregationRule":

1. Дополнительные параметры группировки и стратегии агрегации:
   - Добавлена группировка по дополнительным полям, таким как `user_agent` или `http_method`, для более детального анализа сетевого трафика.
   - Введены новые стратегии агрегации, например, `longest_array` для поля `request_paths`, чтобы отслеживать наиболее часто используемые пути запросов.
2. Изменение параметров `max_events` и `expire_after_ms`:
   - Увеличение значения `max_events` для обработки большего количества событий в одном агрегационном окне.
   - Изменение `expire_after_ms` для оптимизации времени ожидания агрегации событий, учитывая характер трафика.
3. Тестовая проверка:
   - Добавлены тестовые сценарии с учетом новых параметров группировки и стратегий агрегации для проверки правила на способность адекватно агрегировать и анализировать более сложные сетевые события.

Интерпретация правила "EnhancedNetworkActivityAggregationRule"

Рассмотрим механику работы усовершенствованного правила агрегации "EnhancedNetworkActivityAggregationRule", фокусируясь на логике обработки и агрегации событий.

Правило "EnhancedNetworkActivityAggregationRule" предназначено для более глубокого анализа сетевых событий, используя расширенные критерии фильтрации и группировки.

Основные параметры правила:

- `filter`: фильтрация событий основывается на категории "Network", уровне серьезности событий и определенных характеристиках, таких как соответствие `user_agent` шаблону регулярного выражения "Mozilla".
- `group_by`: агрегация событий выполняется по полям `src_ip`, `dest_ip`, `user_agent` и `http_method`, что позволяет проводить многофакторный анализ сетевой активности.
- `starts_when`: условие начала агрегации активируется при обнаружении события начала

соединения или определенных типов HTTP-запросов.

- `merge_strategies`: применяются стратегии агрегации `sum` для полей `bytes_sent` и `bytes_received` и `longest_array` для поля `request_paths`.

Логика агрегации:

- При срабатывании условий `filter` и `starts_when`, события группируются по определенным IP-адресам источника и назначения, типам запросов и агентам пользователя.
- Агрегированные события включают общее количество переданных и полученных байтов (`bytes_sent` и `bytes_received`) и наиболее часто встречающиеся пути запросов (`request_paths`).

Тестирование правила:

Тестовые сценарии "Enhanced Network Activity Test" включают события, соответствующие критериям фильтрации и группировки. Они направлены на проверку способности правила корректно агрегировать данные, учитывая новые параметры группировки и агрегации.

Предположим, что события, поступающие на агрегатор, на котором установлено правило агрегации, имеют следующий вид:

Пример 82. Тестовое событие, JSON

```
{
  "category": "Network",
  "severity": 5,
  "event_type": "connection_start",
  "src_ip": "192.0.2.10",
  "dest_ip": "192.0.2.20",
  "user_agent": "Mozilla/5.0",
  "http_method": "POST",
  "bytes_sent": 2000,
  "bytes_received": 3000
}
```

Тогда, ожидаемый результат агрегации примет следующий вид:

Пример 83. Агрегированное событие, JSON

```
{
  # Служебные поля.
  "id": "e1336694-b3d6-4e02-9f20-92f642bb64fa", # ID агрегированного события.
  "sourceIp": "192.0.2.10", # IP-адрес источника события.
  "tenantId": "99311ef1-32a1-4b2d-9d38-7613be5e4c2b", # ID тенанта.
  "art": "15.03.2024 15:25:05", # Дата и время получения первого события.
  "raw":
"{\"category\":\"Network\",\"severity\":5,\"event_type\":\"connection_start\",\"src_ip
\":\"192.0.2.10\",\"dest_ip\":\"192.0.2.20\",\"user_agent\":\"Mozilla/5.0\",\"http_met
hod\":\"POST\",\"bytes_sent\":2000,\"bytes_received\":3000}", # Текст первого события.
  "collectorId": "891d9a49-8417-4ce9-abf4-184b0c093e46", # ID коллектора.
  "timestamp": "15.03.2024 15:45:10", # Дата и время создания агрегированного события.
```

```
  "aggregationRuleName": "example/network_activity_rule_enhanced", # Название правила
агрегации.
  "aggregationRuleId": "EnhancedNetworkActivityAggregationRule", # ID правила
агрегации.
  "type": "2", # Тип события - "агрегированное событие".
  "cnt": "5", # Количество событий в агрегации.
  "baseEventIds": "41627060-c610-494d-834c-0f6c3223f0eb, b5db4311-b978-45b5-8790-
0b02834bf479, dd3d4ad4-9de9-478e-be65-24eb3e70abd8, 34a9661f-f460-4285-b11b-
8d2388811819, 933a0aaa-85eb-477a-a1a4-285478a42f5a", # Список ID событий в агрегации.
  "baseEventTimestamps": "15.03.2024 15:25:05, 15.03.2024 15:28:35, 15.03.2024
15:33:59, 15.03.2024 15:37:02, 15.03.2024 15:41:17", # Список дат и времен получения
событий в агрегации.
  "groupedBy": "src_ip, dest_ip, user_agent, http_method", # Поля группировки в
агрегации.
  "correlationRuleName": "", # Название корреляционного правила.
  "correlationRuleId": "", # ID корреляционного правила.
  "correlationSeverity": "", # Уровень угрозы кореляционного события.

  # Регулярные поля.
  "category": "Network",
  "event_type": "connection_start",
  "src_ip": "192.0.2.10",
  "dest_ip": "192.0.2.20",
  "user_agent": "Mozilla/5.0",
  "http_method": "POST",
  "total_bytes_sent": 6000,
  "total_bytes_received": 8000,
  "request_paths": "/api/load"
}
```

Подробное описание служебных полей представлено в разделе Универсальная модель события.

Таким образом создается многоуровневая логика агрегации для правила "EnhancedNetworkActivityAggregationRule", где события должны соответствовать заданным критериям фильтрации и группировки. Эти критерии включают не только IP-адреса источника и назначения, но также типы запросов и агенты пользователя, что позволяет проводить более детализированный анализ трафика.

Схемы активных списков

В данном разделе представлен обзор структуры, принципов работы, процесса создания схемы активного списка в системе. Управление активными списками осуществляется в разделе **Активные списки**.

О схемах активных списков

Схема активного списка — это структурированный объект данных в формате RObject (.ro), который задает конфигурацию и параметры для создания и управления активным списком. Схема

определяет структуру данных, правила хранения и управления записями, а также ограничения по количеству и времени жизни записей.

Схема активного списка включает следующие компоненты: идентификатор списка, разметку данных, правила хранения и уникальные ключи для идентификации записей. Поле `capacity` задает максимальное количество записей, которое может быть сохранено в списке, а поле `ttl` (time-to-live) определяет срок хранения каждой записи в секундах. Структура данных описывается с помощью поля `schema`, которое задает типы данных, такие как строки, числа, логические значения и метки времени.

Каждая запись в активном списке идентифицируется по одному или нескольким ключам, указанным в поле `key`, что позволяет гарантировать уникальность записей. Поле `content` содержит фактические данные списка, которые соответствуют структуре, определенной в `schema`.

Работа с активным списком

Доступные операции над активным списком:

- Создание схемы активного списка
- Изменение схемы активного списка
- Удаление схемы активного списка

Создание схемы активного списка

Чтобы добавить схему активного списка:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего списка опцию **Схема активного списка**. Система отобразит окно создания схемы активного списка.

При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля схемы активного списка, чтобы определить логику ее работы.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст схему активного списка и отобразит соответствующее уведомление. Новая схема появится в таблице раздела **Экспертиза**.

Чтобы схема активного списка стала доступна при создании активного списка, ее необходимо включить.

Вы можете создать черновик схемы активного списка без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик схемы и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Изменение схемы активного списка

Чтобы изменить схему активного списка:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку схемы активного списка в списке. Система отобразит в правой части экрана карточку этой схемы с подробной информацией о ней.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек схемы.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

4. Внесите требуемые изменения в конфигурацию схемы активного списка.
5. Нажмите на кнопку **Обновить версию**. Система опубликует новую версию схемы активного списка с измененной конфигурацией и отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию схемы активного списка в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки схемы будет отображаться предупреждение о наличии неопубликованных изменений.

Удаление схемы активного списка

- Удаление доступно только для элементов экспертизы с типом создания **Пользовательский**.
- Если элемент экспертизы используется в других сущностях системы, его удаление невозможно.

Чтобы удалить схему активного списка:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку схемы активного списка в списке. Система отобразит в правой части экрана карточку этой схемы с подробной информацией о ней.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления схемы.
4. Нажмите на кнопку **Удалить**. Система удалит схему активного списка и отобразит

соответствующее уведомление. Удаленная схема будет исключена из списка элементов экспертизы.

Вы также можете удалить группу элементов экспертизы. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные элементы и отобразит соответствующее уведомление. Удаленные элементы будет исключены из списка элементов экспертизы.

Структура схемы активного списка

Схема активного списка в системе содержит набор обязательных и опциональных полей.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор схемы активного списка в системе.	строка	да
`name`	Название схемы списка. Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`), подчеркивания (`_`) и дефис (`-`) без пробелов.	строка	да
`version`	Версия схемы активного списка, представленная в формате Semantic Versioning.	строка	да
`description`	Описание активного списка, где можно указать дополнительную информацию о его назначении и функциональности.	строка	нет
`type`	Тип списка. Всегда имеет значение `active_list` для элемента этого типа.	строка	нет
`author`	Имя и контактные данные автора схемы активного списка.	строка	нет
`data_source`	Список источников данных, с которыми совместима схема активного списка. Используется для классификации и быстрого поиска элементов экспертизы в системе.	массив строк массив объектов	нет
`tags`	Список тегов, присвоенных схеме активного списка для классификации и быстрого поиска.	массив строк	нет

Поле	Описание	Тип данных	Обязательное поле
`capacity`	Максимальное количество записей в активном списке. По достижении лимита, записи удаляются и перезаписываются. Если значение установлено в "0", записи хранятся бессрочно и не перезаписываются.	целое число	да
`ttl`	Время жизни записи в активном списке в секундах. По истечении этого времени запись удаляется. Если значение установлено в "0", запись считается постоянной и хранится бессрочно.	целое число	да
`schema`	Объект, описывающий структуру данных активного списка, включая типы данных для ключей. Поддерживает типы: `string`, `bool`, `float`, `integer`, `timestamp`.	объект	да
`key`	Поле, которое определяет набор полей, которые используются в качестве уникальных идентификаторов в активном списке.	массив строк	да
`content`	Поле содержит массив с фактическими данными активного списка, которые организованы согласно структуре в `schema`.	массив строк	нет

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform> # Платформа или информационная система.
- <source> # Источник событий.
# События (EventId) или путь к журналу событий:
- <EventID_1>
- <EventID_2>
```

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Пример 84. Пример схемы активного списка

```
# Уникальный идентификатор схемы активного списка, необходим для ее идентификации.
```

```
id: unique_identifier/active_list_name

# Название схемы активного списка.
name: ActiveListName

# Версия схемы активного списка.
version: 1.0.0

# Описание схемы активного списка, его назначение и функциональность.
description: Описание схемы активного списка

# Тип элемента экспертизы. Всегда должно быть 'active_list' для элементов такого типа.
type: active_list

# Информация об авторе схемы активного списка.
author: Имя Автора <email@example.com>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_3
    - EventID_22

# Теги для улучшения поиска схемы в системе.
tags:
  - tag1
  - tag2

# Максимальное количество значений в схеме списка. При переполнении удаляется самое
старое значение.
capacity: 10000

# Ключевые поля схемы активного списка должны быть уникальными.
key:
  - field1
  - field2

# Схема элементов в активном списке. Поддерживает типы: 'string', 'bool', 'float',
'integer', 'timestamp'.
schema:
  field1: string
  field2: bool
  field3: float
  field4: integer
  field5: timestamp

# Время жизни значения внутри активного списка, в секундах. По истечении этого времени
значение будет автоматически удалено из списка.
ttl: 0
```

```
# Пример содержимого, соответствующего схеме.
content:
  - field1: "value1"
  - field2: true
  - field3: 1.23
  - field4: 123
  - field5: "2023-01-01T12:00:00Z"
```

Функции управления активными списками

Для работы с записями активного списка в полях элементов экспертизы, которые поддерживают VRL, применяются специальные функции. Доступны функции добавления, получения и удаления записи из активного списка.

Функция	Описание	Параметры
`add_active_record(list, key, value)`	Добавляет запись в активный список. Если запись с таким ключом уже существует, значение обновляется. В противном случае создается новая запись.	• `list`: название схемы активного списка, в который добавляется запись. • `key`: уникальный ключ для записи. • `value`: значение записи.
`get_active_record(list, key)`	Извлекает значение из активного списка по указанному ключу. Возвращает значение, ассоциированное с ключом, или `null`, если запись отсутствует.	• `list`: название схемы активного списка, из которого нужно извлечь запись. • `key`: уникальный ключ записи.
`remove_active_record(list, key)`	Удаляет запись из активного списка. Если запись с таким ключом отсутствует, ничего не происходит.	• `list`: название схемы активного списка, из которого необходимо удалить запись. • `key`: уникальный ключ записи, которую необходимо удалить.

Пример 85. Пример использования в поле `on_correlation` правила корреляции:

```
on_correlation: !vrl |
  if .threat_level == "High" {
    add_active_record!("suspicious_ips", { "ip": .source_ip, "is_suspicious": true })
  }

  if .event_type == "authentication_failure" {
    failure_count = get_active_record!("auth_failure_counts", { "user_id": .user_id
}).failure_count || 0
    add_active_record!("auth_failure_counts", { "user_id": .user_id, {"failure_count":
failure_count + 1 })
  }

  if .event_type == "authentication_success" {
```

```
    remove_active_record("auth_failure_counts", { "user_id": .user_id })
  }
```

В этом примере:

- IP-адреса с высоким уровнем угрозы добавляются в активный список подозрительных IP-адресов.
- Считается количество неудачных попыток аутентификации для пользователей и обновляется соответствующая запись в активном списке.
- При успешной аутентификации запись пользователя удаляется из списка неудачных попыток аутентификации, что может свидетельствовать об окончании атаки или восстановлении доступа пользователя.

Составление схемы активного списка

Управление схемами активных списков в системе осуществляется в разделе **Экспертиза**.

Чтобы составить схему активного списка, необходимо заполнить ее обязательные поля.

Поля метаданных

Заполните поля метаданных схемы активного списка:

- `id`: задайте уникальный идентификатор схемы активного списка.
- `name`: задайте название схемы активного списка. Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`), точки (`.`), подчеркивания (_) и дефис (`-`) без пробелов.
- `version`: укажите версию схемы активного списка в формате Semantic Versioning.
- `type`: укажите `active_list` в поле, чтобы определить тип элемента экспертизы.
- `author` (опциональное): укажите имя и контактные данные автора.
- `description` (опциональное): предоставьте краткое описание назначения и функциональности схемы активного списка.
- `data_source` (опциональное): укажите список совместимых со схемой источников данных для классификации и фильтрации.
- `tags` (опциональное): добавьте теги для классификации и поиска списка в системе.

Пример структуры метаданных:

```
id: "example_whitelist_id"
name: "White list"
version: "1.0.0"
author: "Jane Doe <jane.doe@example.com>"
description: "This static active list contains admin IPs."
data_source:
  - platform: Windows
  - source: Sysmon
tags:
```

```
  - ddos
  - threat intelligence
  - blacklist
```

Поле `capacity`

Поле `capacity` (обязательное) определяет максимальное количество записей, которое может содержать активный список. Это ограничение позволяет контролировать объем хранимых данных и поддерживать производительность системы на оптимальном уровне. По достижении лимита, записи удаляются из списка и перезаписываются. Если значение `capacity` установлено в "0", записи хранятся бессрочно и не перезаписываются.

Пример задания максимального количества записей:

```
capacity: 10000
```

В данном случае, активный список будет ограничен 10000 записями. При достижении этого предела, новые записи будут замещать старые.

Значение по умолчанию для `capacity` не установлено, однако, рекомендуется устанавливать это значение с учетом требований к производительности и объему хранимых данных.

Поле `ttl`

Поле `ttl` (time-to-live) (обязательное) в контексте схемы активного списка определяет максимальное время жизни каждой записи в активном списке, выраженное в секундах. После истечения указанного времени запись автоматически удаляется из списка. Если значение установлено в "0", запись считается постоянной и хранится бессрочно.

Пример:

```
ttl: 86400  # Записи в активном списке будут храниться в течение 24 часов.
```

Установка оптимального значения `ttl` зависит от специфики данных и задач, которые должен выполнять активный список. Для списков с динамически изменяющейся информацией рекомендуется устанавливать `ttl` с учетом ожидаемого времени актуальности данных.

Поле `schema`

Поле `schema` (обязательное) определяет структуру данных активного списка. Это описание включает типы данных для каждого из ключей, которые используются для хранения информации в активном списке. Структура `schema` позволяет определить формат и тип данных каждого элемента списка, обеспечивая тем самым обработку и интерпретацию сохраненной информации.

Схема элементов в активном списке поддерживает типы: `string`, `bool`, `float`, `integer`, `timestamp`.

Структура `schema` и поддерживаемые типы данных:

- `field1`: строка (string)
- `field2`: логический тип (bool)
- `field3`: число с плавающей точкой (float)
- `field4`: целое число (integer)
- `field5`: метка времени (timestamp)

Пример использования в активном списке:

```
schema:
  ip: string        # IP-адрес, представленный в виде строки.
  active: bool      # Признак активности IP-адреса: true или false.
  score: float      # Оценка репутации IP-адреса, выраженная числом с плавающей
точкой.
  attempts: integer # Количество попыток подключения с данного IP-адреса.
  last_seen: timestamp      # Метка времени последнего обнаружения активности с
данного IP-адреса.
  first_detected: timestamp # Метка времени первого обнаружения активности.
```

Поле `keys`

Поле `keys` (обязательное) определяет набор полей, которые используются в качестве уникальных идентификаторов для каждой записи в активном списке. Эти ключи позволяют идентифицировать записи в списке, обеспечивая возможность поиска, обновления или удаления данных. Уникальность ключа гарантирует, что каждая запись в активном списке будет отличаться от других, предотвращая дублирование информации.

Пример:

```
key:
  - ip
```

В данном примере поле `ip` используется в качестве уникального ключа для записей в активном списке. Это означает, что для каждого уникального IP-адреса в списке может быть только одна запись. При попытке добавить новую запись с уже существующим в списке IP-адресом, старая запись будет обновлена или заменена, в зависимости от реализации механизма управления активным списком.

Выбор ключей для активного списка должен основываться на требованиях к уникальности и возможностях идентификации записей. Важно убедиться, что выбранные поля могут обеспечить уникальность записей в контексте предметной области активного списка.

Поле `content`

Поле `content` содержит фактические данные активного списка, представляя собой массив записей, соответствующих определенной схеме данных `schema`. Эта схема определяет структуру и типы данных для каждого элемента в записях, гарантируя таким образом их корректную обработку и хранение.

Пример схемы `schema`, по которой формируется `content`:

```
schema:
  id: integer
  ip: string
  last_detected: timestamp
  attack_type: string
```

Согласно схеме, поле `content` может содержать следующие данные:

```
content:
  - id: 1
    ip: "192.168.0.1"
    last_detected: "2023-09-01T12:00:00Z"
    attack_type: "DDoS"
  - id: 2
    ip: "192.168.0.2"
    last_detected: "2023-09-02T13:00:00Z"
    attack_type: "Phishing"
```

Каждая запись в `content` соответствует структуре и типу данных, определенных в `schema`. В данном примере каждая запись содержит уникальный идентификатор (`id`), IP-адрес (`ip`), дату последнего обнаружения (`last_detected`) и тип атаки (`attack_type`).

Пример схемы активного списка

Пример 86. Пример схемы активного списка "DDoSAttackIPs", схема RObject

```
# Уникальный идентификатор списка.
id: threat_intel/ddos_attack_ips

# Название активного списка, которое обозначает его цель или функцию.
name: DDoSAttackIPs

# Версия активного списка, представленная в формате SemVer (Semantic Versioning).
version: 1.0.0

# Описание активного списка, где можно указать дополнительную информацию о его
назначении и функциональности.
description: Список IP-адресов, обнаруженных как участники в DDoS-атаках, для
мониторинга и предотвращения потенциальных угроз.
```

```
# Тип элемента экспертизы.
type: active_list

# Имя и контактные данные автора активного списка.
author: John Doe <johndoe@example.com>

# Список совместимых с правилом источников данных.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_3
    - EventID_22

# Теги, связанные с активным списком.
tags:
  - ddos
  - threat intelligence
  - blacklist

# Максимальное количество записей в активном списке. Опционально, по умолчанию 100000.
capacity: 10000

# Время жизни записи активного списка в секундах. Опционально, если не указан, то
хранится бессрочно.
ttl: 0

# Схема описывает поля активного списка.
schema:
  ip: string                # IP-адрес.
  lastDetected: timestamp  # Метка времени последнего обнаружения.
  attackType: string       # Тип атаки.
  reputationScore: integer # Оценка репутации IP-адреса.
  attackVolume: string     # Объем атаки.
  involvedAttacks: integer # Количество атак.
  notes: string             # Примечания.

# Перечень полей, которые являются ключом.
key:
  - ip

# Содержимое активного списка.
content:
  - ip: "192.0.2.1"
    lastDetected: "2023-01-01T12:00:00Z"
    attackType: "SYN Flood"
    reputationScore: 10
    attackVolume: "1.5 Gbps"
    involvedAttacks: 5
    notes: "Repeated attacks on multiple targets"
  - ip: "192.0.2.2"
```

```
    lastDetected: "2023-01-02T15:30:00Z"
    attackType: "UDP Flood"
    reputationScore: 20
    attackVolume: "500 Mbps"
    involvedAttacks: 2
    notes: "High packet rate observed"
  - ip: "192.0.2.3"
    lastDetected: "2023-01-03T18:45:00Z"
    attackType: "Amplification"
    reputationScore: 15
    attackVolume: "2 Gbps"
    involvedAttacks: 3
    notes: "DNS amplification attack"
  - ip: "192.0.2.4"
    lastDetected: "2023-01-04T20:00:00Z"
    attackType: "HTTP Flood"
    reputationScore: 5
    attackVolume: "750 Mbps"
    involvedAttacks: 4
    notes: "Targeted at web services"
```

Интерпретация

Схема активного списка "DDoSAttackIPs" служит для учета IP-адресов, обнаруженных как участники в DDoS-атаках.

Метаданные:

- `id`: `threat_intel/ddos_attack_ips` — уникальный идентификатор схемы активного списка.
- `name`: `DDoSAttackIPs` — название схемы активного списка.
- `version`: `1.0.0` — версия схемы, использующаяся для управления изменениями.
- `description`: Схема описывает структуру данных для мониторинга и предотвращения потенциальных угроз от IP-адресов, участвующих в DDoS-атаках.
- `type`: `active_list` — тип элемента экспертизы, указывающий на его предназначение для хранения и обработки информации о потенциальных угрозах.
- `author`: `John Doe <johndoe@rvision.ru>` — информация об авторе схемы, включая контактные данные.
- `tags`: массив тегов (`ddos`, `threat intelligence`, `blacklist`), классифицирующих схему и упрощающих ее поиск и использование в различных контекстах.

Параметры схемы:

- `capacity`: `10000` — максимальное количество записей, которое может содержать активный список, созданный на основе данной схемы.
- `ttl`: `0` — время жизни каждой записи в секундах; значение `0` означает, что записи хранятся бессрочно.
- `key`: массив, указывающий на поля, которые являются ключами для уникальной идентификации

записей в списке; в данном случае это поле `ip`.

Схема данных:

Описывает поля активного списка, включая типы данных и их назначение:

- `ip`: строка, содержащая IP-адрес.
- `lastDetected`: дата, когда IP-адрес был последний раз обнаружен как участник DDoS-атаки.
- `attackType`: строка, описывающая тип DDoS-атаки.
- `reputationScore`: целое число, оценка репутации IP-адреса.
- `attackVolume`: строка, описывающая объем атаки.
- `involvedAttacks`: целое число, количество атак, в которых участвовал данный IP.
- `notes`: строка, содержащая примечания или дополнительную информацию об IP-адресе.

Имена полей могут включать буквы латинского алфавита и цифры (`a–z`, `A–Z`, `0–9`).

Содержимое активного списка:

Примеры записей, которые могут быть добавлены в активный список на основе данной схемы, включают информацию о дате обнаружения, типе атаки, репутационной оценке IP-адреса, объеме атаки, количестве атак, в которых участвовал адрес, и заметках о конкретных атаках.

Схема активного списка "DDoSAttackIPs" используется в активных списках для динамического обновления и хранения информации об IP-адресах, связанных с DDoS-атаками.

Схемы дашбордов

В данном разделе представлен обзор структуры, принципов работы и процесса создания схемы дашборда в системе. Схемы дашбордов используются для настройки конфигурации и параметров отображения данных в дашборде. Управление схемами дашбордов в системе осуществляется в разделе **Экспертиза**.

О схемах дашбордов

Схема дашборда — это структурированный объект данных в формате RObject (.ro), который задает конфигурацию и параметры для создания дашборда. Схема определяет структуру дашборда, правила отображения и обновления данных, а также настройки расположения виджетов.

Схема дашборда включает следующие компоненты: разметку виджетов, параметры каждого виджета и набор переменных для получения данных. Размещение задает позицию, размеры и ограничения для каждого виджета на сетке дашборда. Виджеты отображают данные в различных форматах, таких как графики, таблицы или сводные значения.

Каждый виджет включает описание источника данных, параметры запроса и настройки визуализации. В качестве источников данных могут использоваться метрики или хранилища событий. Например, графики отображают динамику метрик, таблицы — детализированные данные по событиям, а сводные виджеты — итоговые значения.

Схема позволяет настраивать параметры обновления данных, включая частоту и период

отображения. Это обеспечивает возможность отображения как текущих, так и ранее собранных данных. Переменные внутри схемы используются для передачи идентификаторов источников данных и фильтров в запросы.

Работа со схемой дашборда

Доступные операции над схемой дашборда:

- Создание схемы дашборда
- Изменение схемы дашборда
- Удаление схемы дашборда

Создание схемы дашборда

Чтобы создать схему дашборда:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего списка опцию **Схема дашборда**. Система отобразит окно создания схемы.

При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля схемы дашборда, чтобы определить функциональность и внешний вид дашбордов.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст схему дашборда и отобразит соответствующее уведомление. Новая схема появится в таблице раздела **Экспертиза**.

Чтобы схема дашборда стала доступной для создания дашбордов, ее необходимо включить.

Вы можете создать черновик схемы дашборда без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик схемы и отобразит соответствующее уведомление. Новый черновик появится в таблице раздела **Экспертиза**.

Вы можете также создать схему на основе готового дашборда.

Изменение схемы дашборда

Чтобы изменить схему дашборда:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку схемы дашборда в списке. Система отобразит в правой части экрана

карточку этой схемы с подробной информацией о ней.

3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек схемы.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

4. Внесите требуемые изменения в конфигурацию схемы дашборда.
5. Нажмите на кнопку **Обновить версию**. Система сохранит изменения и отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию схемы дашборда в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки схемы будет отображаться предупреждение о наличии неопубликованных изменений.

Удаление схемы дашборда

- Удаление доступно только для элементов экспертизы с типом создания **Пользовательский**.
- Если элемент экспертизы используется в других сущностях системы, его удаление невозможно.

Чтобы удалить схему дашборда:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку схемы дашборда в списке. Система отобразит в правой части экрана карточку этой схемы с подробной информацией о ней.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления схемы.
4. Нажмите на кнопку **Удалить**. Система удалит схему дашборда и отобразит соответствующее уведомление. Удаленная схема будет исключена из списка элементов экспертизы.

Вы также можете удалить группу элементов экспертизы. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.

2. Установите флажки напротив тех элементов, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные элементы и отобразит соответствующее уведомление. Удаленные элементы будет исключены из списка элементов экспертизы.

Структура схемы дашборда

Схема дашборда содержит обязательные и опциональные поля, определяющие его поведение и отображение.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор схемы дашборда. Значение — пять групп из шестнадцатеричных цифр в формате 8-4-4-4-12, разделенных дефисом. Буквы — только строчные латинские. Формат значения: `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.	UUID	да
`name`	Название схемы дашборда. Допустимы латинские буквы, цифры, точки, подчеркивания и дефисы.	строка	да
`description`	Краткое описание схемы дашборда.	строка	нет
`type`	Тип схемы. Для дашбордов всегда указывается `dashboard_schema`.	строка	да
`author`	Имя и контактные данные автора схемы дашборда.	строка	нет
`data_source`	Список источников данных, с которыми совместима схема дашборда. Используется для классификации и быстрого поиска элементов экспертизы в системе по совместимым источникам.	массив строк массив объектов	нет
`tags`	Список тегов, присвоенных схеме дашборда для классификации и быстрого поиска.	массив строк	нет
`version`	Версия схемы дашборда в формате Semantic Versioning.	строка	да
`scope`	Доступность дашборда. Указывает, является ли дашборд персональным (`user`) или общим (`tenant`).	строка	да

Поле	Описание	Тип данных	Обязательное поле
`variables`	Набор переменных, используемых в дашборде для получения данных. Поле `key` — уникальный идентификатор переменной. Поле `type` определяет тип источника данных, с которым связана переменная. Допустимые значения для `type`: `ACTIVE_LIST` — активный список, `EVENT_STORAGE` — хранилище событий, `METRIC` — метрики.	массив объектов	да
`layout`	Разметка виджетов на дашборде, включающая их положение и размеры.	массив объектов	да
`widgets`	Описание виджетов, их настройки и параметры автоматического обновления.	массив объектов	да

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform> # Платформа или информационная система.
- <source> # Источник событий.
# События (EventId) или путь к журналу событий:
- <EventID_1>
- <EventID_2>
```

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `variables`

Поле	Описание	Тип данных	Обязательное поле
`key`	Уникальный идентификатор переменной, используемой в дашборде.	строка	да
`type`	Тип источника данных, с которым связана переменная. Допустимые значения: `event_storage` для хранилища событий, `metric` для метрик.	строка	да

Структура `layout`

Поле `layout` описывает расположение виджетов на дашборде, включая их позицию на сетке и размеры.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор виджета.	строка	да
`x`, `y`	Позиция виджета на сетке (`x` — по горизонтали, `y` — по вертикали).	число	да
`w`, `h`	Размеры виджета в ячейках сетки (ширина `w`, высота `h`).	число	да
`maxH`, `minH`	Максимальная и минимальная высота виджета.	число	да
`minW`	Минимальная ширина виджета.	число	да

Структура `widgets`

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор виджета. Используется для его ссылки в других элементах схемы дашборда.	строка	да
`name`	Название виджета, которое будет отображаться на дашборде.	строка	да
`description`	Краткое описание виджета, поясняющее его назначение.	строка	нет
`autoUpdate`	Настройки автоматического обновления данных виджета. Включает единицу времени (`unit`) и значение интервала обновления (`value`).	объект	да
`last`	Указывает временной период данных, которые будут отображаться на виджете. Например, за последний час или месяц.	объект	нет
`variables`	Переменные, которые используются для запроса данных виджета. Включает источник данных (`source`) и дополнительные параметры запроса.	объект	да

Структура `autoUpdate`

Поле	Описание	Тип данных	Обязательное поле
`unit`	Единица времени, через которую будет происходить автоматическое обновление данных виджета. Допустимые значения: `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `YEAR`.	строка	да
`value`	Интервал обновления в единицах, указанных в поле `unit`. Например, при значении `unit: MINUTE` и `value: 5`, обновление будет происходить каждые 5 минут.	число	да

Структура `last`

Поле	Описание	Тип данных	Обязательное поле
`unit`	Единица времени, за которую будут отображаться данные. Допустимые значения: `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `YEAR`.	строка	да
`value`	Числовое значение интервала для выбранной единицы времени. Например, при значении `unit: DAY` и `value: 7` будут отображены данные за последние 7 дней.	число	да

Структура `variables`

Поле	Описание	Тип данных	Обязательное поле
`source`	Определяет источник данных для виджета. Допустимые значения: `metric`, `search`, `alert`.	строка	да
`metricId`	Идентификатор метрики, данные которой должны быть отображены. Применяется, если `source` установлен на `metric`.	строка	да (если `source` = `metric`)
`chartType`	Тип графика, который используется для визуализации данных. Допустимые значения: `line_chart`, `bar_chart`, `pie_chart`, `table`, `total`.	строка	да (если используется график)
`rql`	Запрос на языке RQL, который используется для получения данных. Применяется, если `source` установлен на `search`.	строка	да (если `source` = `search`)
`nameMapping`	Массив, который задает сопоставление полей данных с отображаемыми элементами (например, `x`, `y` для графиков или `timestamp`, `cnt` для таблиц).	массив объектов	да (для графиков и таблиц)

Составление схемы дашборда

Для создания схемы дашборда необходимо определить ряд обязательных и опциональных полей, которые формируют структуру и поведение дашборда.

Поля метаданных

Заполните поля метаданных схемы дашборда:

- `id`: уникальный идентификатор схемы дашборда. Значение — пять групп из шестнадцатеричных цифр в формате 8-4-4-4-12, разделенных дефисом. Буквы — только строчные латинские. Формат значения: `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.
- `name`: название схемы дашборда. Допустимы латинские буквы, цифры, символы подчеркивания, точки и дефисы.
- `type`: всегда указывается `dashboard_schema` для элементов экспертизы данного типа.
- `description` (опционально): краткое описание назначения дашборда.
- `author` (опционально): имя и контактные данные автора схемы дашборда.
- `version`: версия схемы дашборда согласно стандарту Semantic Versioning.
- `scope`: доступность дашборда. Указывает, является ли он персональным (`user`) или общим (`tenant`).
- `data_source` (опционально): список источников данных, с которыми совместима схема дашборда. Используется для классификации и быстрого поиска элементов экспертизы в системе.
- `tags` (опционально): список тегов, присвоенных схеме дашборда для классификации и быстрого поиска.

Пример метаданных:

```
id: "61f0c404-5cb3-11e7-907b-000000000000"
name: "System Monitoring Dashboard"
type: dashboard_schema
description: "Дашборд для мониторинга системных метрик и событий"
author: John Doe <johndoe@rvision.ru>
version: "0.0.1"
scope: tenant
data_source:
  - platform: Windows
  - source: Sysmon
tags:
  - monitoring
  - system
```

В этом примере:

- Схема дашборда с уникальным идентификатором `61f0c404-5cb3-11e7-907b-000000000000` предназначена для мониторинга системных метрик и событий.
- Название схемы `System Monitoring Dashboard` описывает назначение дашборда.

- Поле `type` указывает на тип элемента экспертизы (`dashboard_schema`).
- Версия схемы дашборда задана как `0.0.1` в соответствии с правилами семантического версионирования.
- Автором схемы является `John Doe`, чьи контактные данные указаны в поле `author`.
- В поле `data_source` перечислены источники данных, совместимые с дашбордом — `Windows` и `Sysmon`.
- Теги `monitoring` и `system` присвоены для облегчения поиска и классификации дашборда в системе.

Поле `variables`

Поле `variables` используется для передачи переменных в дашборд. Переменные могут включать следующие типы источников данных: хранилища событий и метрики.

- `key`: уникальный идентификатор переменной.
- `type`: тип источника данных (допустимые значения: `event_storage` для хранилища событий, `metric` для метрик).

Пример:

```
variables:
  - key: VARIABLE_EVENT_STORAGE_ID_1
    type: event_storage
```

В этом примере:

- Переменная с ключом `VARIABLE_EVENT_STORAGE_ID_1` используется для обозначения хранилища событий в дашборде. Тип переменной установлен как `event_storage`, что указывает на использование хранилища событий в запросах данных.

Поле `layout`

Поле `layout` определяет размещение виджетов на дашборде. Каждый виджет имеет уникальный идентификатор и параметры, определяющие его положение и размеры:

- `id`: уникальный идентификатор виджета.
- `x`, `y`: координаты виджета на сетке (`x`-- позиция по горизонтали, `y` — позиция по вертикали).
- `w`, `h`: размеры виджета в ячейках (ширина `w` и высота `h`).
- `maxH`, `minH`, `minW`: ограничения по минимальной и максимальной высоте и ширине.

Пример разметки виджетов:

```
layout:
  - id: WIDGET_KEY_1
    x: 0
    y: 1
```

```
    w: 6
    h: 3
    maxH: 5
    minH: 3
    minW: 6
  - id: WIDGET_KEY_2
    x: 0
    y: 4
    w: 6
    h: 3
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_1` размещен в позиции `x: 0`, `y: 1`, что означает, что он будет располагаться в первом столбце и второй строке сетки дашборда.
- Его ширина составляет 6 ячеек (`w: 6`), а высота — 3 ячейки (`h: 3`).
- Для этого виджета также установлены ограничения: максимальная высота (`maxH`) — 5 ячеек, минимальная высота (`minH`) — 3 ячейки и минимальная ширина (`minW`) — 6 ячеек.
- Виджет с идентификатором `WIDGET_KEY_2` размещен в позиции `x: 0`, `y: 4`, что означает, что он будет находиться в первом столбце и на пятой строке сетки дашборда.
- Его размеры такие же, как у первого виджета — ширина 6 ячеек и высота 3 ячейки. Ограничения по высоте и ширине в данном примере для второго виджета не заданы.

Поле `widgets`

Поле `widgets` описывает каждый виджет, его данные и тип отображаемой информации. Основные параметры виджетов:

- `id`: уникальный идентификатор виджета.
- `name`: название виджета.
- `description`: краткое описание функциональности виджета.
- `autoUpdate`: настройки автоматического обновления данных (единица времени и интервал).
- `last`: период отображаемых данных (например, последний месяц или день).
- `variables`: параметры запроса данных для виджета.

Пример описания виджета:

```
widgets:
  - id: WIDGET_KEY_1
    name: "Host DB RAM"
    description: "Отображает текущее использование ОЗУ базы данных"
    autoUpdate:
      unit: MINUTE
      value: 1
    last:
      unit: DAY
```

```
      value: 1
    variables:
      source: metric
      metricId: "c9c9e971-7d45-48eb-9ed5-000000000000"
      queryArguments: []
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_1` назван "Host DB RAM" и предназначен для отображения текущего использования оперативной памяти базы данных.
- Поле `description` описывает функциональность виджета как "Отображает текущее использование ОЗУ базы данных".
- Виджет настроен на автоматическое обновление данных каждые 1 минуту, что задается параметрами в поле `autoUpdate` (`unit: MINUTE` и `value: 1`).
- Поле `last` указывает, что виджет будет отображать данные за последний день (`unit: DAY` и `value: 1`).
- В поле `variables` указано, что данные берутся из источника типа `metric`, с идентификатором метрики "c9c9e971-7d45-48eb-9ed5-000000000000". Дополнительные параметры запроса определены как `queryArguments: []`.

Структура виджетов

Виджеты на дашборде могут быть нескольких типов в зависимости от отображаемых данных и формата представления. Каждый тип виджета имеет уникальные параметры.

Виджеты с метриками

Метрики отображают количественные данные, такие как использование ресурсов системы, количество событий и другие параметры, собираемые из источников.

Поля виджета с метриками:

- `source`: определяет источник данных для виджета (допустимые значения: `metric`, `search`, `alert`).
- `metricId`: идентификатор метрики, данные которой отображаются.
- `step`: интервал времени, за который агрегируются данные.

Пример виджета с метрикой:

```
widgets:
  - id: WIDGET_KEY_1
    name: "CPU Usage"
    description: "Отображает использование CPU системы"
    autoUpdate:
      unit: MINUTE
      value: 5
    last:
      unit: HOUR
```

```
      value: 1
    variables:
      source: metric
      metricId: "c9c9e971-7d45-48eb-9ed5-000000000000"
      step: ONE_MIN
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_1` называется "CPU Usage" и предназначен для отображения текущей загрузки процессора системы.
- Поле `description` содержит описание "Отображает использование CPU системы".
- Виджет обновляется каждые 5 минут, что задано в поле `autoUpdate` (`unit: MINUTE`, `value: 5`).
- Поле `last` указывает, что данные за последний час будут агрегироваться и отображаться на виджете (`unit: HOUR`, `value: 1`).
- Источник данных для виджета определен в поле `variables`. Здесь указывается, что данные берутся из метрики (`source: metric`), с идентификатором метрики `metricId: "c9c9e971-7d45-48eb-9ed5-000000000000"`, а интервал агрегации данных установлен на одну минуту (`step: ONE_MIN`).

Виджеты с графиками

Графики визуализируют изменение данных во времени и поддерживают несколько типов отображения: линейные графики, столбчатые диаграммы, круговые диаграммы и др.

Поля виджета с графиком:

- `chartType`: тип графика (`line_chart`, `bar_chart`, `pie_chart`, `table`, `total`).
- `rql`: запрос на языке RQL для получения данных.
- `nameMapping`: сопоставление полей запроса с осями графика (например, `x` — время, `y` — количество).
- `limit`: лимит строк для виджета.
- `searchEntity`: идентификатор переменной и тип источника данных.

Пример виджета с линейным графиком:

```
widgets:
  - id: WIDGET_KEY_2
    name: "Event Storage Counter"
    description: "Отображает количество событий в хранилище"
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: MONTH
      value: 1
    variables:
      source: search
```

```
      chartType: line_chart
      rql: "SELECT count(id) as cnt, toStartOfHour(timestamp) as ts GROUP BY ts"
      nameMapping:
        - key: x
          value: ts
        - key: y
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_2` называется "Event Storage Counter" и предназначен для отображения количества событий в хранилище.
- Поле `description` содержит описание "Отображает количество событий в хранилище".
- Виджет автоматически обновляется каждые 30 минут, что задано в поле `autoUpdate` (`unit: MINUTE`, `value: 30`).
- Поле `last` указывает, что виджет будет отображать данные за последний месяц (`unit: MONTH`, `value: 1`).
- В поле `variables` виджет использует источник данных типа `search` и тип графика `line_chart`, что означает, что данные будут отображаться в виде линейного графика.
- Запрос на языке RQL (`rql`) возвращает количество событий (`cnt`), сгруппированных по часам (`toStartOfHour(timestamp)`).
- Поле `nameMapping` используется для сопоставления полей запроса с осями графика: поле `ts` отображается на оси `x`, а количество событий `cnt` — на оси `y`.
- Поле `limit` определяет лимит в 10 000 строк.
- Поле `searchEntity` указывает имя переменной, которая соответствует хранилищу событий.

Табличные виджеты

Табличные виджеты представляют данные в виде таблицы, показывая несколько столбцов с данными.

Поля табличного виджета:

- `chartType`: всегда имеет значение `TABLE`.
- `rql`: запрос на языке RQL для получения данных, которые будут отображены в таблице.
- `nameMapping`: сопоставление полей со столбцами таблицы (например, `timestamp`, `count`).
- `limit`: лимит строк для виджета.
- `searchEntity`: идентификатор переменной и тип источника данных.

Пример табличного виджета:

```
widgets:
  - id: WIDGET_KEY_3
    name: "Event Table"
    description: "Отображает события с временными метками"
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: DAY
      value: 1
    variables:
      source: search
      chartType: table
      rql: "SELECT timestamp, count(id) as cnt FROM event_table GROUP BY timestamp"
      nameMapping:
        - key: timestamp
          value: timestamp
        - key: count
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_3` называется "Event Table" и отображает события с указанием временных меток.
- Поле `description` содержит описание "Отображает события с временными метками".
- Виджет автоматически обновляется каждые 30 минут (`autoUpdate` с параметрами `unit: MINUTE` и `value: 30`).
- Поле `last` указывает, что виджет будет отображать данные за последний день (`unit: DAY`, `value: 1`).
- В поле `variables` указано, что виджет использует тип отображения данных `table` (`chartType: TABLE`), а данные запрашиваются из источника типа `search`.
- Запрос на языке RQL (`rql`) получает временные метки (`timestamp`) и количество событий (`cnt`), сгруппированных по времени.
- Поле `nameMapping` сопоставляет поля запроса со столбцами таблицы: поле `timestamp` отображается в столбце с временными метками, а поле `cnt` — в столбце с количеством событий.
- Поле `limit` определяет лимит в 10 000 строк.
- Поле `searchEntity` указывает имя переменной, которая соответствует хранилищу событий.

Виджеты со сводными данными

Сводные виджеты отображают итоговые значения по выбранным параметрам, такие как общее количество событий или использования ресурсов.

Поля сводного виджета:

- `chartType`: всегда имеет значение `total`.
- `rql`: запрос на языке RQL для получения сводных данных.
- `nameMapping`: сопоставление результата с отображаемым полем.
- `limit`: лимит строк для виджета.
- `searchEntity`: идентификатор переменной и тип источника данных.

Пример сводного виджета:

```
widgets:
  - id: WIDGET_KEY_5
    name: "Total Event Count"
    description: "Отображает общее количество событий"
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: MONTH
      value: 1
    variables:
      source: search
      chartType: total
      rql: "SELECT count(id) as cnt FROM event_storage"
      nameMapping:
        - key: value
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_5` назван "Total Event Count" и отображает общее количество событий.
- Поле `description` содержит описание "Отображает общее количество событий".
- Виджет автоматически обновляется каждые 30 минут (`autoUpdate` с параметрами `unit: MINUTE` и `value: 30`).
- Поле `last` указывает, что виджет будет отображать данные за последний месяц (`unit: MONTH`, `value: 1`).
- Источник данных — хранилище событий (`source: search`), тип виджета — сводный (`chartType: total`).
- Запрос на языке RQL (`rql`) возвращает общее количество событий в хранилище (`SELECT count(id) as cnt FROM event_storage`).
- Поле `nameMapping` сопоставляет поле `cnt` с отображаемым полем виджета, обозначенным как

`value`.

- Поле `limit` определяет лимит в 10 000 строк.
- Поле `searchEntity` указывает имя переменной, которая соответствует хранилищу событий.

Пример схемы дашборда

Пример 87. Пример схемы дашборда "System Monitoring Dashboard", схема RObject

```
# Универсальный уникальный идентификатор схемы дашборда.
id: 61f0c404-5cb3-11e7-907b-000000000000

# Название дашборда.
name: System Monitoring Dashboard

# Тип элемента экспертизы.
type: dashboard_schema

# Версия схемы.
version: 1.0.0

# Принадлежность дашборда (личный или общий).
scope: user

# Краткое описание назначения дашборда.
description: Дашборд для мониторинга системных метрик и событий в реальном времени

# Имя и контактные данные автора схемы.
author: John Doe <johndoe@example.com>

# Источники данных, с которыми совместима схема дашборда.
data_source:
  - platform: Windows
  - source: Sysmon
  - events:
    - EventID_1
    - EventID_2

# Теги для классификации и быстрого поиска схемы.
tags:
  - monitoring
  - system

# Переменные для источников данных виджетов.
variables:
  - key: VARIABLE_EVENT_STORAGE_ID_1
    type: EVENT_STORAGE

# Размещение виджетов на дашборде.
layout:
  - id: WIDGET_KEY_1
```

```
    x: 0
    y: 1
    w: 6
    h: 3
    maxH: 5
    minH: 3
    minW: 6
  - id: WIDGET_KEY_2
    x: 0
    y: 4
    w: 6
    h: 3
  - id: WIDGET_KEY_3
    x: 0
    y: 7
    w: 6
    h: 3

# Параметры виджетов на дашборде.
widgets:
  - id: WIDGET_KEY_1
    name: Host DB RAM
    description: Виджет для отображения текущего использования оперативной памяти
базой данных
    autoUpdate:
      unit: MINUTE
      value: 1
    last:
      unit: DAY
      value: 1
    variables:
      source: metric
      metricId: "c9c9e971-7d45-48eb-9ed5-000000000000"
      queryArguments: []

  - id: WIDGET_KEY_2
    name: Event Storage Counter (LineChart)
    description: Линейный график для отображения количества событий в хранилище за
месяц
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: MONTH
      value: 1
    variables:
      source: search
      chartType: line_chart
      rql: "SELECT count(id) as cnt, toStartOfHour(timestamp) as ts GROUP BY ts"
      nameMapping:
        - key: x
```

```
          value: ts
        - key: y
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage


- id: WIDGET_KEY_3
    name: Event Storage Table
    description: Таблица для отображения событий из хранилища с указанием времени и
количества событий
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: DAY
      value: 1
    variables:
      source: search
      chartType: table
      rql: "SELECT timestamp, count(id) as cnt FROM event_table GROUP BY timestamp"
      nameMapping:
        - key: timestamp
          value: timestamp
        - key: count
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

Интерпретация

Метаданные схемы дашборда:

- `id`: универсальный уникальный идентификатор схемы дашборда (UUID) — "61f0c404-5cb3-11e7-907b-000000000000".
- `name`: название схемы дашборда — "System Monitoring Dashboard".
- `type`: тип элемента — всегда указывается как `dashboard_schema`.
- `version`: версия схемы — "1.0.0".
- `scope`: принадлежность дашборда — "user", что указывает на то, что дашборд доступен только конкретному пользователю.
- `description`: краткое описание назначения дашборда — "Дашборд для мониторинга системных метрик и событий в реальном времени".
- `author`: автор схемы дашборда — "John Doe <johndoe@example.com>".

- `data_source`: источники данных, с которыми совместима схема — "Windows", "Sysmon".
- `tags`: теги, присвоенные схеме для классификации — "monitoring", "system".

Переменные:

- `variables`: переменные используются для передачи данных в виджеты. В данном примере переменная `VARIABLE_EVENT_STORAGE_ID_1` определена как идентификатор хранилища событий.

Разметка виджетов:

- `layout`: разметка задает размещение виджетов на дашборде. Например, виджет с идентификатором `WIDGET_KEY_1` размещен в позиции `x: 0`, `y: 1` с шириной 6 и высотой 3.

Виджеты:

1. **Host DB RAM**: отображает использование оперативной памяти базы данных. Виджет обновляется каждые 1 минуту, и отображаются данные за последний день.
   - `source`: метрика (используется источник данных типа `metric`).
   - `metricId`: идентификатор метрики для получения данных о RAM — "c9c9e971-7d45-48eb-9ed5-000000000000".
2. **Event Storage Counter (LineChart)**: линейный график для отображения количества событий в хранилище за последний месяц. Обновление происходит каждые 30 минут.
   - `source`: данные из хранилища событий (используется источник `search`).
   - `chartType`: тип графика — линейный (`line_chart`).
   - `rql`: запрос на языке RQL для получения количества событий и их группировки по часам.
   - `nameMapping`: сопоставление полей запроса с осями графика (ось `x` — время `ts`, ось `y` — количество событий `cnt`).
   - `limit`: лимит строк (10 000).
3. **Event Storage Table**: таблица для отображения событий с указанием времени и количества событий. Обновление происходит каждые 30 минут.
   - `source`: данные из хранилища событий (используется источник `search`).
   - `chartType`: тип отображения — таблица (`table`).
   - `rql`: запрос на языке RQL для группировки событий по времени и подсчета их количества.
   - `nameMapping`: сопоставление полей для столбцов таблицы (столбец `timestamp` — время события, столбец `cnt` — количество событий).
   - `limit`: лимит строк (10 000).

Схемы шаблонов отчетов

В данном разделе представлен обзор структуры, принципов работы и процесса создания схемы шаблона отчета в системе. Схемы шаблонов отчетов используются для настройки конфигурации и параметров формирования шаблонов отчетов. Управление схемами шаблонов отчетов в системе осуществляется в разделе **Экспертиза**.

О схемах шаблонов отчетов

Схема шаблона отчета — это структурированный объект данных в формате RObject (.ro), который задает конфигурацию и параметры для создания шаблона отчета. Схема определяет структуру отчета, правила его формирования и отображения данных.

Схема шаблона отчета включает следующие компоненты: параметры хранения данных, расписание формирования отчета, разметку и параметры виджетов, а также переменные для получения данных. Параметры хранения задают период времени, в течение которого данные остаются доступными для включения в отчеты. Разметка определяет расположение виджетов, которые визуализируют данные в различных форматах, таких как графики, таблицы и сводные значения.

Каждый виджет содержит источник данных, параметры обновления и запросы для получения информации. Данные могут поступать из различных источников, таких как метрики или хранилища событий. Переменные используются для передачи идентификаторов источников данных и других параметров в запросы, что позволяет управлять выборкой данных для отчета.

Работа со схемой шаблона отчета

Доступные операции над схемой шаблона отчета:

- Создание схемы шаблона отчета
- Изменение схемы шаблона отчета
- Удаление схемы шаблона отчета

Создание схемы шаблона отчета

Чтобы добавить схему шаблона отчета:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего списка опцию **Схема шаблона отчета**. Система отобразит окно создания схемы.

При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля схемы, чтобы определить функциональность и внешний вид отчетов.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст схему шаблона отчета и отобразит соответствующее уведомление. Новая схема появится в таблице раздела **Экспертиза**.

Чтобы схема стала доступной для создания шаблонов отчета, ее необходимо включить.

Вы можете создать черновик схемы шаблона без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик схемы и отобразит соответствующее уведомление. Новый

черновик появится в таблице раздела **Экспертиза**.

Вы можете также создать схему на основе готового шаблона отчета.

Изменение схемы шаблона отчета

Чтобы изменить схему шаблона отчета:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку схемы шаблона отчета в списке. Система отобразит в правой части экрана карточку этой схемы с подробной информацией о ней.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек схемы.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

4. Внесите требуемые изменения в конфигурацию схемы шаблона отчета.
5. Нажмите на кнопку **Обновить версию**. Система опубликует новую версию схемы шаблона отчета с измененной конфигурацией и отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию схемы шаблона отчета в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Система сохранит конфигурацию для дальнейшего редактирования и отобразит соответствующее уведомление. В верхней части карточки схемы будет отображаться предупреждение о наличии неопубликованных изменений.

Удаление схемы шаблона отчета

- Удаление доступно только для элементов экспертизы с типом создания **Пользовательский**.
- Если элемент экспертизы используется в других сущностях системы, его удаление невозможно.

Чтобы удалить схему шаблона отчета:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку схемы шаблона отчета в списке. Система отобразит в правой части экрана

карточку этой схемы с подробной информацией о ней.

3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления схемы.
4. Нажмите на кнопку **Удалить**. Система удалит схему шаблона отчета и отобразит соответствующее уведомление. Удаленная схема будет исключена из списка элементов экспертизы.

Вы также можете удалить группу элементов экспертизы. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные элементы и отобразит соответствующее уведомление. Удаленные элементы будет исключены из списка элементов экспертизы.

Структура схемы шаблона отчета

Схема шаблона отчета содержит обязательные и опциональные поля, которые определяют его параметры и функциональность.

Поле	Описание	Тип данных	Обязательное поле
`id`	Универсальный уникальный идентификатор схемы шаблона отчета. Значение — пять групп из шестнадцатеричных цифр в формате 8-4-4-4-12, разделенных дефисом. Буквы — только строчные латинские. Формат значения: `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.	UUID	да
`name`	Название шаблона отчета. Допустимы латинские буквы, цифры, точки, подчеркивания и дефисы.	строка	да
`description`	Краткое описание шаблона отчета.	строка	нет
`type`	Тип схемы. Для шаблонов отчетов всегда указывается `report_template_schema`.	строка	да
`author`	Имя и контактные данные автора схемы шаблона отчета.	строка	нет
`data_source`	Список источников данных, с которыми совместима схема шаблона отчета. Используется для классификации и быстрого поиска элементов экспертизы в системе по совместимым источникам.	массив строк массив объектов	нет

Поле	Описание	Тип данных	Обязательное поле
`tags`	Список тегов, присвоенных схеме шаблона отчета для классификации и быстрого поиска.	массив строк	нет
`version`	Версия схемы шаблона отчета в формате Semantic Versioning.	строка	да
`storagePeriod`	Параметры времени хранения данных, включаемых в отчет.	объект	да
`schedule`	Параметры расписания формирования отчетов согласно формату Cron.	объект	да
`variables`	Набор переменных, используемых в отчете для получения данных. Поле `key` — уникальный идентификатор переменной. Поле `type` определяет тип источника данных, с которым связана переменная. Допустимые значения для `type`: `active_list` — активный список, `event_storage` — хранилище событий, `metric` — метрики.	массив объектов	да
`layout`	Размещение виджетов в отчете, определяющая их расположение и размеры.	массив объектов	да
`widgets`	Описание виджетов, их настройки и параметры автоматического обновления.	массив объектов	да
`autoUpdate`	Настройки автоматического обновления данных виджета. Включает единицу времени (`unit`), допустимые значения: `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `YEAR`, и значение интервала обновления (`value`) в виде положительного целого числа.	объект	да

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform> # Платформа или информационная система.
- <source> # Источник событий.
# События (EventId) или путь к журналу событий:
- <EventID_1>
- <EventID_2>
```

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `storagePeriod`

Поле `storagePeriod` определяет временной интервал, в течение которого данные остаются доступными для формирования отчетов.

Поле	Описание	Тип данных	Обязательное поле
`unit`	Единица времени, например, `MINUTE`, `HOUR`, `DAY`, `MONTH`.	строка	да
`value`	Числовое значение для указанной единицы времени. Например, при значении `unit: MINUTE` и `value: 60`, данные будут храниться 60 минут.	число	да

Структура `schedule`

Поле `schedule` задает параметры расписания согласно формату Cron для автоматического формирования отчетов.

Поле	Описание	Тип данных	Обязательное поле
`hours`	Часы, в которые будет формироваться отчет. Указывается в формате `0-23`.	строка	да
`minutes`	Минуты, в которые будет формироваться отчет. Указывается в формате `0-59`.	строка	да
`seconds`	Секунды для формирования отчета. Указывается в формате `0-59`.	строка	да
`dayOfMonth`	День месяца для формирования отчета. Указывается в формате `1-31` или `*` для ежедневного формирования.	строка	да
`dayOfWeek`	День недели для формирования отчета. Указывается в формате `0-6` (где 0 — это воскресенье) или `*` для ежедневного формирования.	строка	да
`month`	Месяц для формирования отчета. Указывается в формате `1-12` или `*` для формирования отчета каждый месяц.	строка	да

Структура `variables`

Поле	Описание	Тип данных	Обязательное поле
`key`	Уникальный идентификатор переменной, используемой в шаблоне отчета.	строка	да
`type`	Тип источника данных, с которым связана переменная. Допустимые значения: `event_storage` для хранилища событий, `metric` для метрик.	строка	да

Структура `layout`

Поле `layout` описывает размещение виджетов в отчете, включая их позицию на сетке и размеры.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор виджета.	строка	да
`x`, `y`	Позиция виджета на сетке (`x` — по горизонтали, `y` — по вертикали).	число	да
`w`, `h`	Размеры виджета в ячейках сетки (ширина `w`, высота `h`).	число	да
`maxH`, `minH`	Максимальная и минимальная высота виджета.	число	да
`minW`	Минимальная ширина виджета.	число	да

Структура `widgets`

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор виджета. Используется для его ссылки в других элементах схемы шаблона отчета.	строка	да
`name`	Название виджета, которое будет отображаться в отчете.	строка	да
`description`	Краткое описание виджета, поясняющее его назначение.	строка	нет
`autoUpdate`	Настройки автоматического обновления данных виджета. Включает единицу времени (`unit`) и значение интервала обновления (`value`).	объект	да
`last`	Указывает временной период данных, которые будут отображаться на виджете. Например, за последний час или месяц.	объект	нет

Поле	Описание	Тип данных	Обязательное поле
`variables`	Переменные, которые используются для запроса данных виджета. Включает источник данных (`source`) и дополнительные параметры запроса.	объект	да

Структура `autoUpdate`

Поле	Описание	Тип данных	Обязательное поле
`unit`	Единица времени, через которую будет происходить автоматическое обновление данных виджета. Допустимые значения: `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `YEAR`.	строка	да
`value`	Интервал обновления в единицах, указанных в поле `unit`. Например, при значении `unit: MINUTE` и `value: 5`, обновление будет происходить каждые 5 минут.	число	да

Структура `last`

Поле	Описание	Тип данных	Обязательное поле
`unit`	Единица времени, за которую будут отображаться данные. Допустимые значения: `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `YEAR`.	строка	да
`value`	Числовое значение интервала для выбранной единицы времени. Например, при значении `unit: DAY` и `value: 7` будут отображены данные за последние 7 дней.	число	да

Структура `variables`

Поле	Описание	Тип данных	Обязательное поле
`source`	Определяет источник данных для виджета. Допустимые значения: `metric`, `search`.	строка	да
`metricId`	Идентификатор метрики, данные которой должны быть отображены. Применяется, если `source` установлен на `metric`.	строка	да (если `source` = `metric`)
`chartType`	Тип графика, который используется для визуализации данных. Допустимые значения: `LINE_CHART`, `BAR_CHART`, `PIE_CHART`, `TABLE`, `TOTAL`.	строка	да (если используется график)

Поле	Описание	Тип данных	Обязательное поле
`rql`	Запрос на языке RQL, который используется для получения данных. Применяется, если `source` установлен на `search`.	строка	да (если `source` = `search`)
`nameMapping`	Массив, который задает сопоставление полей данных с отображаемыми элементами (например, `x`, `y` для графиков или `timestamp`, `cnt` для таблиц).	массив объектов	да (для графиков и таблиц)

Составление схемы шаблона отчета

Для создания схемы шаблона отчета необходимо определить ряд обязательных и опциональных полей, которые формируют структуру и поведение шаблона отчета.

Поля метаданных

Заполните поля метаданных схемы шаблона отчета:

- `id`: уникальный идентификатор схемы шаблона отчета. Значение — пять групп из шестнадцатеричных цифр в формате 8-4-4-4-12, разделенных дефисом. Буквы — только строчные латинские. Формат значения: `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`. Должен быть уникальным в рамках коллектора.
- `name`: название схемы шаблона отчета. Допустимы латинские буквы, цифры, символы подчеркивания, точки и дефисы.
- `description` (опционально): краткое описание назначения шаблона отчета.
- `type`: всегда указывается `report_template_schema` для элементов экспертизы данного типа.
- `version`: версия схемы шаблона отчета согласно стандарту Semantic Versioning.
- `author`: имя и контактные данные автора схемы шаблона отчета.
- `data_source`: список источников данных, с которыми совместима схема шаблона отчета. Это поле используется для классификации и быстрого поиска элементов в системе.
- `tags`: список тегов, присвоенных схеме шаблона отчета, для классификации и упрощения поиска.

Пример метаданных:

```
id: "61f0c404-5cb3-11e7-907b-000000000000"
name: "System Report Template"
description: "Шаблон для создания отчетов по метрикам системы и событиям"
type: report_template_schema
version: "1.0.0"
author: John Doe <johndoe@example.com>
data_source:
  - Windows
  - Linux
tags:
  - system
```

```
  - report
```

В этом примере:

- Схема шаблона отчета с уникальным идентификатором `61f0c404-5cb3-11e7-907b-000000000000` предназначена для создания отчетов по метрикам системы и событиям.
- Название схемы `System Report Template` описывает назначение шаблона.
- Поле `type` указывает, что тип элемента экспертизы (`report_template_schema`).
- Версия шаблона отчета задана как `1.0.0` в соответствии с правилами семантического версионирования.
- Автором схемы является `John Doe`, чьи контактные данные указаны в поле `author`.
- В поле `data_source` перечислены источники данных, совместимые с отчетом — `Windows` и `Linux`.
- Теги `system` и `report` присвоены для облегчения поиска и классификации шаблона отчета в системе.

Поле `storagePeriod`

Поле `storagePeriod` определяет временной интервал, в течение которого данные остаются доступными для включения в отчет.

- `unit`: единица времени, например, `MINUTE`, `HOUR`, `DAY`, `MONTH`.
- `value`: числовое значение для выбранной единицы времени.

Пример:

```
storagePeriod:
  unit: MINUTE
  value: 60
```

В этом примере:

- Поле `storagePeriod` задает временной интервал для сохранения данных, который равен 60 минутам. Это означает, что данные, собранные за последние 60 минут, будут включены в отчет.

Поле `schedule`

Поле `schedule` задает параметры расписания согласно формату Cron для автоматического формирования отчетов.

- `hours`: часы, в которые будет формироваться отчет (формат `0-23`).
- `minutes`: минуты для формирования отчета (формат `0-59`).
- `seconds`: секунды для формирования отчета (формат `0-59`).
- `dayOfMonth`: день месяца для формирования отчета (`1-31` или `*` для ежедневного отчета).
- `dayOfWeek`: день недели для формирования отчета (`0-6`, где 0 — это воскресенье).
- `month`: месяц для формирования отчета (`1-12` или `*` для ежемесячного отчета).

Пример:

```
schedule:
  hours: "10"
  minutes: "0"
  seconds: "0"
  dayOfMonth: "*"
  dayOfWeek: "*"
  month: "*"
```

В этом примере:

- Отчет будет формироваться каждый день в 10:00:00 утра, так как `hours` установлено на "10", а `minutes` и `seconds` установлены на "0".
- Поле `dayOfMonth` установлено на `*`, что означает формирование отчета каждый день месяца.
- Поля `dayOfWeek` и `month` также установлены на `*`, что задает ежедневное и ежемесячное формирование отчета без ограничений по конкретным дням недели или месяцам.

Поле `variables`

Поле `variables` используется для передачи переменных в шаблон отчета.

- `key`: уникальный идентификатор переменной.
- `type`: тип источника данных (допустимые значения: `event_storage` для хранилища событий, `metric` для метрик).

Пример:

```
variables:
  - key: VARIABLE_EVENT_STORAGE_ID_1
    type: event_storage
```

В этом примере:

- Переменная с ключом `VARIABLE_EVENT_STORAGE_ID_1` используется для обозначения хранилища событий в шаблоне отчета. Тип переменной установлен как `event_storage`, что указывает на использование хранилища событий в запросах данных.

Поле `layout`

Поле `layout` определяет размещение виджетов в шаблоне отчета. Каждый виджет имеет уникальный идентификатор и параметры, определяющие его положение и размеры.

- `id`: уникальный идентификатор виджета.
- `x`, `y`: координаты виджета на сетке (`x` — позиция по горизонтали, `y` — по вертикали).
- `w`, `h`: размеры виджета в ячейках (ширина `w` и высота `h`).
- `maxH`, `minH`, `minW`: ограничения по минимальной и максимальной высоте и ширине.

Пример:

```
layout:
  - id: WIDGET_KEY_1
    x: 0
    y: 1
    w: 6
    h: 3
    maxH: 5
    minH: 3
    minW: 6
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_1` размещен в позиции `x: 0`, `y: 1`, что означает его расположение в первом столбце и второй строке сетки шаблона отчета.
- Размер виджета составляет 6 ячеек по ширине (`w: 6`) и 3 ячейки по высоте (`h: 3`).
- Ограничения для виджета установлены: максимальная высота — 5 ячеек, минимальная высота — 3 ячейки, минимальная ширина — 6 ячеек.

Поле `widgets`

Поле `widgets` описывает каждый виджет, его параметры данных и тип отображаемой информации. Основные параметры виджетов:

- `id`: уникальный идентификатор виджета.
- `name`: название виджета.
- `description`: краткое описание функциональности виджета.
- `autoUpdate`: настройки автоматического обновления данных (единица времени и интервал).
- `last`: временной период отображаемых данных (например, последний день или месяц).
- `variables`: параметры запроса данных для виджета.

Пример:

```
widgets:
  - id: WIDGET_KEY_1
    name: "Host DB RAM"
    description: "Отображает текущее использование RAM базой данных"
    autoUpdate:
      unit: MINUTE
      value: 1
    last:
      unit: DAY
      value: 1
    variables:
      source: metric
      metricId: c9c9e971-7d45-48eb-9ed5-000000000000
```

```
      queryArguments: []
      step: ONE_DAY
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_1` имеет название "Host DB RAM" и предназначен для отображения использования оперативной памяти базы данных.
- Описание виджета указано в поле `description` как "Отображает текущее использование RAM базой данных".
- Виджет настроен на автоматическое обновление каждые 1 минуту, что задано в поле `autoUpdate`.
- Поле `last` указывает, что виджет будет отображать данные за последний день.
- Источник данных для виджета — метрика (`source: metric`), с идентификатором метрики `metricId: c9c9e971-7d45-48eb-9ed5-000000000000`. Дополнительные параметры запроса определены как `queryArguments: []`, и шаг агрегации данных установлен на один день (`step: ONE_DAY`).

Структура виджетов

Виджеты в шаблоне отчета могут быть нескольких типов в зависимости от отображаемых данных и формата представления. Каждый тип виджета имеет уникальные параметры.

Виджеты с метриками

Метрики отображают количественные данные, такие как использование ресурсов системы, количество событий и другие параметры, собираемые из источников.

Поля виджета с метриками:

- `source`: определяет источник данных для виджета (допустимые значения: `metric`, `SEARCH`, `alert`).
- `metricId`: идентификатор метрики, данные которой отображаются.
- `step`: интервал времени, за который агрегируются данные.

Пример виджета с метрикой:

```
widgets:
  - id: WIDGET_KEY_1
    name: "CPU Usage"
    description: "Отображает использование CPU системы"
    autoUpdate:
      unit: MINUTE
      value: 5
    last:
      unit: HOUR
      value: 1
    variables:
      source: metric
      metricId: "c9c9e971-7d45-48eb-9ed5-000000000000"
```

```
      step: ONE_MIN
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_1` называется "CPU Usage" и предназначен для отображения текущей загрузки процессора системы.
- Поле `description` содержит описание "Отображает использование CPU системы".
- Виджет обновляется каждые 5 минут, что задано в поле `autoUpdate` (`unit: MINUTE`, `value: 5`).
- Поле `last` указывает, что данные за последний час будут агрегироваться и отображаться на виджете (`unit: HOUR`, `value: 1`).
- Источник данных для виджета определен в поле `variables`. Здесь указывается, что данные берутся из метрики (`source: metric`), с идентификатором метрики `metricId: "c9c9e971-7d45-48eb-9ed5-000000000000"`, а интервал агрегации данных установлен на одну минуту (`step: ONE_MIN`).

Виджеты с графиками

Графики визуализируют изменение данных во времени и поддерживают несколько типов отображения: линейные графики, столбчатые диаграммы, круговые диаграммы и др.

Поля виджета с графиком:

- `chartType`: тип графика (`line_chart`, `bar_chart`, `pie_chart`, `table`, `total`).
- `rql`: запрос на языке RQL для получения данных.
- `nameMapping`: сопоставление полей запроса с осями графика (например, `x` — время, `y` — количество).
- `limit`: лимит строк для виджета.
- `searchEntity`: идентификатор переменной и тип источника данных.

Пример виджета с линейным графиком:

```
widgets:
  - id: WIDGET_KEY_2
    name: "Event Storage Counter"
    description: "Отображает количество событий в хранилище"
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: MONTH
      value: 1
    variables:
      source: search
      chartType: line_chart
      rql: "SELECT count(id) as cnt, toStartOfHour(timestamp) as ts GROUP BY ts"
      nameMapping:
        - key: x
```

```
          value: ts
        - key: y
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_2` называется "Event Storage Counter" и предназначен для отображения количества событий в хранилище.
- Поле `description` содержит описание "Отображает количество событий в хранилище".
- Виджет автоматически обновляется каждые 30 минут, что задано в поле `autoUpdate` (`unit: MINUTE`, `value: 30`).
- Поле `last` указывает, что виджет будет отображать данные за последний месяц (`unit: MONTH`, `value: 1`).
- В поле `variables` виджет использует источник данных типа `search` и тип графика `line_chart`, что означает, что данные будут отображаться в виде линейного графика.
- Запрос на языке RQL (`rql`) возвращает количество событий (`cnt`), сгруппированных по часам (`toStartOfHour(timestamp)`).
- Поле `nameMapping` используется для сопоставления полей запроса с осями графика: поле `ts` отображается на оси `x`, а количество событий `cnt` — на оси `y`.
- Поле `limit` определяет лимит в 10 000 строк.
- Поле `searchEntity` указывает имя переменной, которая соответствует хранилищу событий.

Табличные виджеты

Табличные виджеты представляют данные в виде таблицы, показывая несколько столбцов с данными.

Поля табличного виджета:

- `chartType`: всегда имеет значение `table`.
- `rql`: запрос на языке RQL для получения данных, которые будут отображены в таблице.
- `nameMapping`: сопоставление полей со столбцами таблицы (например, `timestamp`, `count`).
- `limit`: лимит строк для виджета.
- `searchEntity`: идентификатор переменной и тип источника данных.

Пример табличного виджета:

```
widgets:
  - id: WIDGET_KEY_3
    name: "Event Table"
    description: "Отображает события с временными метками"
```

```
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: DAY
      value: 1
    variables:
      source: search
      chartType: table
      rql: "SELECT timestamp, count(id) as cnt FROM event_table GROUP BY timestamp"
      nameMapping:
        - key: timestamp
          value: timestamp
        - key: count
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_3` называется "Event Table" и отображает события с указанием временных меток.
- Поле `description` содержит описание "Отображает события с временными метками".
- Виджет автоматически обновляется каждые 30 минут (`autoUpdate` с параметрами `unit: MINUTE` и `value: 30`).
- Поле `last` указывает, что виджет будет отображать данные за последний день (`unit: DAY`, `value: 1`).
- В поле `variables` указано, что виджет использует тип отображения данных `table` (`chartType: TABLE`), а данные запрашиваются из источника типа `search`.
- Запрос на языке RQL (`rql`) получает временные метки (`timestamp`) и количество событий (`cnt`), сгруппированных по времени.
- Поле `nameMapping` сопоставляет поля запроса со столбцами таблицы: поле `timestamp` отображается в столбце с временными метками, а поле `cnt` — в столбце с количеством событий.
- Поле `limit` определяет лимит в 10 000 строк.
- Поле `searchEntity` указывает имя переменной, которая соответствует хранилищу событий.

Виджеты со сводными данными

Сводные виджеты отображают итоговые значения по выбранным параметрам, такие как общее количество событий или использования ресурсов.

Поля сводного виджета:

- `chartType`: всегда имеет значение `total`.
- `rql`: запрос на языке RQL для получения сводных данных.

- `nameMapping`: сопоставление результата с отображаемым полем.
- `limit`: лимит строк для виджета.
- `searchEntity`: идентификатор переменной и тип источника данных.

Пример сводного виджета:

```
widgets:
  - id: WIDGET_KEY_5
    name: "Total Event Count"
    description: "Отображает общее количество событий"
    autoUpdate:
      unit: MINUTE
      value: 30
    last:
      unit: MONTH
      value: 1
    variables:
      source: search
      chartType: total
      rql: "SELECT count(id) as cnt FROM event_storage"
      nameMapping:
        - key: value
          value: cnt
      limit: 10000
      searchEntity:
        id: VARIABLE_EVENT_STORAGE_ID_1
        type: event_storage
```

В этом примере:

- Виджет с идентификатором `WIDGET_KEY_5` назван "Total Event Count" и отображает общее количество событий.
- Поле `description` содержит описание "Отображает общее количество событий".
- Виджет автоматически обновляется каждые 30 минут (`autoUpdate` с параметрами `unit: MINUTE` и `value: 30`).
- Поле `last` указывает, что виджет будет отображать данные за последний месяц (`unit: MONTH`, `value: 1`).
- Источник данных — хранилище событий (`source: search`), тип виджета — сводный (`chartType: total`).
- Запрос на языке RQL (`rql`) возвращает общее количество событий в хранилище (`SELECT count(id) as cnt FROM event_storage`).
- Запрос на языке RQL (`rql`) возвращает общее количество событий в хранилище (`SELECT count(id) as cnt FROM event_storage`).
- Поле `nameMapping` сопоставляет поле `cnt` с отображаемым полем виджета, обозначенным как `value`.

- Поле `limit` определяет лимит в 10 000 строк.
- Поле `searchEntity` указывает имя переменной, которая соответствует хранилищу событий.

Таблицы обогащения

В данном разделе освещается концепция таблиц обогащения в системе, включая описание их назначения, принципы работы и процесса интеграции в процесс обработки событий. Управление таблицами обогащения происходит в разделе **Экспертиза**.

О таблицах обогащения

Таблица обогащения в системе представляет собой структурированный набор данных, используемый для дополнения и уточнения информации по событиям безопасности. Они служат для обогащения событий дополнительными данными, не содержащимися непосредственно в событиях, но позволяющими улучшить анализ и интерпретацию событий.

Структура таблиц обогащения включает в себя определения полей и типов данных, которые должны содержаться в каждой записи таблицы. Каждая запись в таблице обогащения может включать такие данные, как идентификаторы устройств, IP-адреса, имена пользователей, географические данные, данные о приложениях и любую другую информацию, которая может быть использована для обогащения событий.

Процесс обогащения событий с использованием таблиц обогащения заключается в сопоставлении данных из событий с записями в таблицах по определенным ключам. Это может быть выполнено автоматически системой во время обработки событий, когда обнаруживается совпадение по ключу, и дополнительные данные из таблицы включаются в информацию о событии.

Таблицы обогащения позволяют выполнить следующие задачи:

- Дополнение событий информацией, недоступной в исходных данных, для улучшения понимания контекста события.
- Уточнение и классификация событий на основе дополнительных параметров, содержащихся в таблицах.
- Повышение точности корреляции событий за счет дополнительных данных для сопоставления.

Для управления таблицами обогащения предусмотрены функции обогащения для получения и добавления записей, а также механизмы для автоматического обновления данных в таблицах с использованием внешних источников.

Работа с таблицей обогащения

Доступные операции над таблицей обогащения:

- Создание таблицы обогащения
- Изменение таблицы обогащения
- Удаление таблицы обогащения

Создание таблицы обогащения

Чтобы добавить таблицу обогащения:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего опцию **Таблица обогащения**. Система отобразит окно создания таблицы обогащения.

При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля таблицы обогащения, чтобы определить логику ее работы.
4. Нажмите на кнопку **Опубликовать версию**, чтобы сохранить изменения и опубликовать созданную таблицу обогащения. Система создаст таблицу обогащения и отобразит соответствующее уведомление. Новая таблица появится в списке элементов экспертизы.

Чтобы таблица обогащения стала доступна на конвейерах событий, ее необходимо включить.

Вы можете создать черновик таблицы обогащения без публикации с помощью кнопки **Сохранить черновик**. Система создаст черновик таблицы обогащения и отобразит соответствующее уведомление. Новый черновик появится в списке элементов экспертизы.

Изменение таблицы обогащения

Чтобы изменить таблицу обогащения:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку таблицы обогащения в списке. Система отобразит в правой части экрана карточку этой таблицы с подробной информацией о ней.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек таблицы.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

4. Внесите требуемые изменения в конфигурацию таблицы обогащения.
5. Нажмите на кнопку **Обновить версию**. Система опубликует новую версию таблицы обогащения с

измененной конфигурацией и отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию таблицы обогащения в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки таблицы будет отображаться предупреждение о наличии неопубликованных изменений.

Удаление таблицы обогащения

- Удаление доступно только для элементов экспертизы с типом создания **Пользовательский**.
- Если элемент экспертизы используется в других сущностях системы, его удаление невозможно.

Чтобы удалить таблицу обогащения:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку таблицы обогащения в списке. Система отобразит в правой части экрана карточку этой таблицы с подробной информацией о ней.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления таблицы.
4. Нажмите на кнопку **Удалить**. Система удалит таблицу обогащения и отобразит соответствующее уведомление. Удаленная таблица будет исключена из списка.

Вы также можете удалить группу элементов экспертизы. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные элементы и отобразит соответствующее уведомление. Удаленные элементы будет исключены из списка элементов экспертизы.

Структура таблицы обогащения

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор таблицы обогащения.	строка	да

Поле	Описание	Тип данных	Обязательное поле
`name`	Название таблицы обогащения. Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`), подчеркивания (_) и дефис (`-`) без пробелов.	строка	да
`type`	Тип элемента. Всегда имеет значение `enrichment_table` для элементов этого типа.	строка	да
`version`	Версия таблицы обогащения, представленная в формате Semantic Versioning.	строка	да
`author`	Имя и контактные данные автора таблицы обогащения.	строка	нет
`description`	Описание таблицы обогащения, содержащее дополнительную информацию о ее назначении и функциональности.	строка	нет
`tags`	Список тегов, присвоенных таблице обогащения для классификации и быстрого поиска.	массив строк	нет
`kind`	Тип хранилища таблицы обогащения, указывающий на формат (`table`, `maxmind`).	строка	да
`location`	Способ коммуникации с хранилищем таблицы обогащения (`file`, `embedded`).	строка	да
`schema`	Схема, описывающая поля таблицы обогащения и их типы данных. Поддерживаются типы `string`, `bool`, `float`, `integer`, `date`, `timestamp`.	объект	да
`content`	Содержимое таблицы для режима `embedded`, включающее предопределенные данные. Обязательно только для таблиц с `location: embedded`.	массив объектов	условно (требуется только для `embedded`)
`update_frequency_min`	Частота обновления данных в минутах для режимов, отличных от `embedded`. Определяет, как часто данные должны обновляться из внешнего источника.	число	условно (для `file`)
`uri`	Расположение файла или источника данных.	строка	условно (для `file`)

Структура `tests`:

Поле	Описание	Тип данных	Обязательное поле
`name`	Название теста, которое позволяет уникально идентифицировать его результаты в выводе тестирования.	строка	да
`events`	Набор тестовых событий, имитирующих входные данные, которые будут обрабатываться правилом. Это позволяет проверить реакцию правила на разнообразные ситуации.	массив	да
`assertion`	Блок утверждений на языке VRL, применяемых к результатам обработки тестовых событий правилом. Функции `assert` и `assert_eq` используются для проверки соответствия фактических результатов обработки ожидаемым значениям.	строка	да

Эта структура позволяет определить тесты для проверки функциональности таблицы обогащения, включая корректность обработки данных и соответствие результатов обогащения заданным условиям.

Составление таблицы обогащения

Управление таблицами обогащения в системе осуществляется в разделе **Экспертиза**.

Для создания таблицы обогащения в системе необходимо определить ее ключевые элементы структуры. Эти элементы охватывают как метаданные таблицы, так и параметры, определяющие содержание и методы доступа к данным.

1. Определите метаданные таблицы:
    - `id`: уникальный идентификатор таблицы обогащения.
    - `name`: название таблицы обогащения. Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`), точки (`.`), подчеркивания (_) и дефис (`-`) без пробелов.
    - `version`: версия таблицы, следующая стандартам Semantic Versioning.
    - `type`: тип элемента, указываемый как `enrichment_table`.
    - `author` (опционально): имя и контактные данные автора таблицы.
    - `description` (опционально): краткое описание назначения и функциональности таблицы.
    - `tags` (опционально): теги для классификации и поиска таблицы.
2. Настройте параметры доступа и структуру данных:
    - `kind`: тип хранилища данных (например, `table`, `maxmind`).
    - `location`: метод доступа к данным (например, `file`, `embedded`).
    - `schema`: схема данных, определяющая структуру и типы полей таблицы.

    - `content` (для `embedded`): содержимое таблицы, предоставленное непосредственно в конфигурации для встроенного хранения.
3. Установите параметры обновления данных (для режимов, отличных от `embedded`):
    - `update_frequency_min`: частота обновления данных в минутах.
    - `uri` (для `file`): расположение файла или источника данных.

Поля метаданных

Метаданные служат для идентификации и описания таблицы обогащения в системе. Заполнение метаданных должно производиться в соответствии со следующими указаниями:

- `id`: присвойте уникальный идентификатор таблицы обогащения. Идентификатор должен быть уникальным в пределах системы.
- `name`: укажите название таблицы обогащения. Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`), точки (`.`), подчеркивания (_) и дефис (`-`) без пробелов.
- `version`: определите версию таблицы, следуя принципам Semantic Versioning (например, 1.0.0).
- `type`: укажите тип элемента как `enrichment_table`.
- `author` (опциональное): укажите имя и контактные данные автора таблицы.
- `description` (опциональное): предоставьте краткое описание целей и функционала таблицы обогащения.
- `tags` (опциональное): добавьте теги для удобства классификации и поиска таблицы в системе.

Поле `kind`

Поле `kind` в таблицах обогащения определяет тип источника данных, используемого для обогащения информации. Это поле позволяет системе определить, каким образом должны быть загружены и обработаны данные, а также какие операции поддерживаются для данного типа таблицы. Важность этого поля заключается в его влиянии на методы доступа к данным и способы их использования в процессе обогащения.

Типы источников данных

- `table`: указывает, что данные для обогащения хранятся в структурированной таблице, которая может быть загружена из файла, получена через API или встроена непосредственно в конфигурацию.
- `maxmind`: определяет использование специализированных геолокационных баз данных MaxMind для обогащения данных, например, для определения местоположения по IP-адресу.

Выбор типа источника данных

Выбор между `table` и `maxmind` зависит от конкретных задач обогащения и доступности данных. Для геолокационного обогащения чаще всего используется тип `maxmind`, тогда как для других видов обогащения может быть выбран тип `table`.

Пример использования поля `kind`:

```
kind: table
```

или

```
kind: maxmind
```

Выбор типа `kind` должен соответствовать целям обогащения и типу обрабатываемых данных. Важно учитывать, что выбор типа источника данных влияет на конфигурацию таблицы обогащения и ее дальнейшее использование в системе.

При планировании использования таблиц обогащения рекомендуется заранее определить требуемый тип источника данных, чтобы обеспечить оптимальную конфигурацию и эффективность процесса обогащения.

Поле `location`

Поле `location` в таблицах обогащения указывает на метод хранения и доступа к данным таблицы обогащения. Этот параметр критически важен для определения того, как система будет взаимодействовать с данными, включая их загрузку, обновление и использование в процессе обогащения.

Значения поля `location` и их описание

- `file`: данные таблицы обогащения располагаются в файле. Система обращается к файлу для чтения данных. Путь к файлу указывается отдельно в конфигурации.
- `embedded`: данные встроены непосредственно в конфигурацию таблицы обогащения. Этот метод подходит для статических данных, которые не требуют регулярных обновлений.

Выбор метода `location`

Пример:

```
location: file
```

или

```
location: embedded
```

Поле `schema`

Поле `schema` в таблицах обогащения определяет структуру данных, которая будет использоваться в таблице. Оно задает типы данных для каждого поля в таблице обогащения, обеспечивая корректную обработку и интерпретацию данных при их использовании.

Определите поля данных. Каждое поле в `schema` должно иметь указание своего типа данных, что позволяет системе правильно обрабатывать входящие значения.

Доступные типы данных:

- `string` — для текстовой информации,
- `bool` — для булевых значений (истина/ложь),
- `float` — для чисел с плавающей точкой,
- `integer` — для целочисленных значений,
- `date` — для дат,
- `timestamp` — для меток времени.

Пример:

```
schema:
  user_id: integer
  user_email: string
  login_time: timestamp
  is_active: bool
```

Используйте `schema` для точного определения ожидаемых данных. Это включает в себя не только типы полей, но и их назначение. Например, поле `user_email` должно содержать адрес электронной почты пользователя, а `login_time` — временную метку его последнего входа в систему.

Поле `content`

Поле `content` используется для хранения данных непосредственно в конфигурации таблицы, когда выбран режим `location: embedded`. Это поле позволяет интегрировать статические данные в таблицу обогащения без необходимости их загрузки из внешних источников. `content` обеспечивает немедленный доступ к данным для обогащения без задержек, связанных с их извлечением или обновлением.

Структура и заполнение поля `content`

`content` является массивом объектов, где каждый объект представляет собой запись данных в таблице. Структура каждого объекта должна соответствовать схеме, определенной в поле `schema`.

Пример:

```
content:
  - id: 1
    name: "Example Name 1"
    type: "Type1"
  - id: 2
    name: "Example Name 2"
    type: "Type2"
```

Использование `content` требует тщательного планирования структуры данных и их соответствия схеме таблицы. Данные должны быть актуальными и достоверными, поскольку они будут встроены в таблицу без возможности динамического обновления (за исключением ручного изменения конфигурации). `content` подходит для данных, которые не изменяются или обновляются редко, таких как списки контрольных значений, статические справочники и прочие.

Рекомендации по заполнению

- Перед включением данных в `content` убедитесь, что они точно соответствуют определенной в `schema` структуре. Некорректно структурированные данные могут привести к ошибкам при их обработке.
- Для удобства управления и возможного обновления рассмотрите возможность разделения больших объемов данных на несколько таблиц обогащения, особенно если данные относятся к различным категориям или имеют разное назначение.

Включение данных в `content` увеличивает независимость от внешних источников и сетевой инфраструктуры, однако требует аккуратности в подготовке и проверке этих данных для обеспечения корректности обогащения.

Поле `update_frequency_min`

Поле `update_frequency_min` определяет периодичность обновления данных в таблице обогащения в минутах. Этот параметр актуален для таблиц, метод доступа к данным которых определен как `file` в поле `location`. Значение поля позволяет настроить частоту, с которой система будет инициировать процесс обновления данных, чтобы поддерживать их актуальность.

Принципы использования

- Значение `update_frequency_min` должно быть выбрано с учетом баланса между необходимостью актуальности данных и минимизацией нагрузки на системные ресурсы.
- Для данных, подверженных частым изменениям, рекомендуется устанавливать меньшее значение, обеспечивая тем самым их своевременное обновление.
- Для статических данных, которые изменяются редко, целесообразно выбирать большее значение интервала обновления.

Пример:

```
update_frequency_min: 30
```

В данном примере данные в таблице обогащения будут обновляться каждые 30 минут.

Рекомендации по заполнению

- Оцените частоту изменений в данных источника и потребности бизнес-процессов в актуальной информации при настройке `update_frequency_min`.
- Учитывайте возможности инфраструктуры и избегайте чрезмерной нагрузки на систему,

оптимизируя частоту обновлений.

Настройка `update_frequency_min` для каждой таблицы обогащения отдельно позволяет точнее контролировать актуальность данных в соответствии с их спецификой и обновлениями. Использование разных интервалов обновления дает возможность более гибкого реагирования на потребности в данных, оптимизируя ресурсы и улучшая эффективность обогащения.

Поле `uri`

Поле `uri` в таблицах обогащения указывает на расположение файла или источника данных, из которого таблица будет получать обновления. Это поле обязательно для метода доступа `file`, поскольку оно определяет, где именно хранятся данные и как они будут извлекаться.

Пример:

```
http://10.0.0.1/test.csv
```

Функции обогащения

Функции обогащения используются для поиска и извлечения данных из таблиц обогащения. Эти функции позволяют обогатить события безопасности дополнительной информацией, не содержащейся непосредственно в событиях. Взаимодействие с таблицами обогащения может быть настроено в любом поле, использующем язык VRL.

Для взаимодействия с таблицами обогащения предусмотрены VRL функции поиска значений в таблице обогащения `find_enrichment_table_records` и извлечения значения из таблицы обогащения `get_enrichment_table_record`.

Пример 88. Функция поиска значений в таблице обогащения

```
find_enrichment_table_records(таблица: <строка>, условие: <объект>, [select: <массив>,
case_sensitive: <логическое значение>])
:: <массив>
```

Поиск записей в таблице обогащения, которые соответствуют заданным условиям. Возвращает массив объектов, соответствующих критериям поиска.

Пример 89. Функция извлечения значения из таблицы обогащения

```
get_enrichment_table_record(table: <строка>, condition: <объект>, [select: <массив>,
case_sensitive: <логическое значение>])
:: <объект>, <ошибка>
```

Извлекает единственную запись из таблицы обогащения, точно соответствующую указанным условиям. В случае, если запись не найдена или найдено более одной записи, возвращается ошибка.

Функция	Описание	Возвращаемый тип
`find_enrichment_table_records`	Поиск множества записей в таблице обогащения по заданным критериям.	Массив объектов
`get_enrichment_table_record`	Поиск одной конкретной записи, точно соответствующей условиям поиска.	Объект или ошибка

Пример 90. Пример использования функции `find_enrichment_table_records`:

```
find_enrichment_table_records("geoip", {"ip": .source_ip})
```

Данная функция ищет записи в таблице обогащения `geoip`, соответствующие IP-адресу источника. Возвращает массив объектов, каждый из которых содержит информацию о геолокации по IP-адресу.

Пример 91. Пример использования функции `get_enrichment_table_record`:

```
get_enrichment_table_record("user_profiles", {"user_id": .user_id})
```

Функция ищет в таблице обогащения `user_profiles` запись, точно соответствующую идентификатору пользователя. Возвращает объект с профилем пользователя или ошибку, если запись не найдена или найдено несколько записей.

Пример 92. Пример использования в поле `on_correlate`:

```
on_correlate: !vrl |
  let geo_info = find_enrichment_table_records("geoip", {"ip": .source_ip}, select:
["country_name", "city_name"])
  if !geo_info.is_empty() {
    .source_country = geo_info[0].country_name
    .source_city = geo_info[0].city_name
  }

  let threat_info = get_enrichment_table_record("threat_intelligence", {"ip":
.source_ip}, select: ["category", "score"])
  if !is_null(threat_info) {
    .threat_category = threat_info.category
    .threat_score = threat_info.score
  }
```

В этом примере:

- Выполняется поиск геолокационной информации по IP-адресу источника события в таблице `geoip`, и результаты добавляются к событию как `.source_country` и `.source_city`.
- Затем производится запрос к таблице `threat_intelligence` для извлечения информации о категории угрозы и оценке уровня угрозы, ассоциированных с IP-адресом источника. Эти данные добавляются к событию как `.threat_category` и `.threat_score`.

Пример таблицы обогащения

Пример 93. Пример таблицы обогащения "DDoSSuspiciousPatterns", схема RObject

```
# Уникальный идентификатор таблицы обогащения.
id: security/ddos_suspicious_patterns

# Название таблицы обогащения.
name: DDoSSuspiciousPatterns

# Тип элемента экспертизы.
type: enrichment_table

# Версия таблицы.
version: 1.0.0

# Описание назначения таблицы.
description: Содержит список подозрительных паттернов запросов и IP-адресов,
ассоциированных с DDoS атаками.

# Имя и контактные данные автора таблицы.
author: John Doe <johndoe@rvision.ru>

# Теги, связанные с таблицей.
tags:
  - ddos
  - security

# Способ коммуникации с хранилищем таблицы.
location: embedded

# Тип хранилища таблицы.
kind: table

# Схема описывает поля таблицы обогащения.
schema:
  ip_address: string
  pattern: string
  risk_level: integer

# Содержимое таблицы для `embedded` режима.
content:
  - ip_address: "192.168.1.1"
    pattern: "/api/highload"
    risk_level: 5
  - ip_address: "192.168.100.100"
    pattern: "/api/highload"
    risk_level: 5
  - ip_address: "172.16.0.2"
    pattern: "/contact"
    risk_level: 2
```

```
  - ip_address: "192.168.100.132"
    pattern: "/api/highload"
    risk_level: 5
```

Интерпретация

Метаданные таблицы обогащения:

- `id`: "security/ddos_suspicious_patterns" – уникальный идентификатор, позволяющий идентифицировать таблицу в системе.
- `name`: "DDoSSuspiciousPatterns" – название таблицы, отражающее ее предназначение для хранения данных о подозрительных паттернах запросов, связанных с DDoS атаками.
- `type`: "enrichment_table" – указывает на тип объекта в системе, подтверждая его принадлежность к таблицам обогащения.
- `version`: "1.0.0" – версия таблицы обогащения, следует принципам Semantic Versioning, обеспечивая управление изменениями и совместимостью.
- `description`: описывает назначение таблицы, в данном случае содержание списка подозрительных паттернов запросов и IP-адресов, ассоциированных с DDoS атаками.
- `author`: "John Doe <johndoe@rvision.ru>" – информация об авторе таблицы, включая контактные данные.
- `tags`: ["ddos", "security"] – теги, упрощающие поиск и классификацию таблицы в системе.

Параметры доступа и хранения данных:

- `location: embedded` – указывает на то, что данные таблицы встроены непосредственно в конфигурацию, что упрощает доступ к ним и ускоряет процесс обогащения.
- `kind`: "table" – подтверждает, что данные организованы в виде структурированной таблицы, подходящей для поиска и обработки информации.

Структура данных (схема и содержимое):

- `schema` описывает структуру данных в таблице, включая поля `ip_address` (строка), `pattern` (строка) и `risk_level` (целое число), что позволяет систематически хранить информацию о подозрительных IP-адресах, соответствующих паттернах запросов и уровне риска.
- `content` содержит конкретные записи данных, например, IP-адреса с ассоциированными с ними паттернами запросов и уровнем риска, демонстрируя реальные примеры данных, используемых для обогащения событий.

Обогащение событий:

- Когда событие содержит IP-адрес или паттерн запроса, система проверяет наличие соответствующих записей в таблице "DDoSSuspiciousPatterns".

Сопоставление событий с записями в таблице:

- Для каждого события система ищет записи в таблице, где `ip_address` совпадает с IP-адресом источника события или `pattern` совпадает с паттерном запроса, указанным в событии.

- Если сопоставление успешно, информация о `risk_level` из таблицы добавляется к событию, указывая на уровень риска, связанный с обнаруженным IP-адресом или паттерном запроса.

Шаблоны поиска

Шаблоны поиска в системе используются для настройки конфигурации и условий RQL-запросов. Они применяются в разделе **Поиск** и виджетах дашбордов для отображения аналитических данных.

Управление конфигурациями шаблонов поиска осуществляется в разделе **Экспертиза**. Шаблоны позволяют задавать фильтры, временные интервалы и RQL-запросы для выборки событий, упрощая анализ данных и создание визуализаций.

О шаблонах поиска

Шаблон поиска — это структурированный объект данных в формате RObject (.ro), который определяет параметры и условия для выполнения запросов к данным в системе в разделе **Поиск** и виджетах дашбордов. Их можно создавать на основе запросов, введенных в этих разделах, или непосредственно через раздел Экспертиза. Шаблон поиска задает структуру фильтров, временной интервал и параметры выборки, которые могут быть использованы для выполнения поисковых запросов по событиям системы.

Шаблон поиска включает следующие компоненты: фильтры для настройки условий отбора данных, временные параметры и RQL-запрос, формирующий логику поиска. Фильтры определяют поля и условия, по которым данные отбираются, временные параметры указывают период выборки, а запрос на языке RQL задает основную логику обработки данных.

Каждый фильтр включает описание целевого поля данных, логический оператор сравнения и значения, по которым данные должны отбираться. В качестве условий могут использоваться операторы `EQ`, `BETWEEN`, `LIKE`, `IN`, `GT`, `GE`, `LT`, `LE`, и `NE`.

Шаблон поиска позволяет настраивать временные параметры, включая фиксированный период и интервальные значения. Это обеспечивает возможность как статических, так и динамических выборок данных на основе шаблона. Поля фильтров и временных настроек передаются в RQL-запрос, который используется для конечной выборки данных.

Работа с шаблоном поиска

Шаблоны поиска используются в разделе **Поиск**.

Доступные операции над шаблоном поиска:

- Создание шаблона
- Применение шаблона
- Поиск шаблона
- Обновление шаблона

Создание шаблона

Чтобы создать шаблон:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Нажмите на кнопку **Создать шаблон**. Отобразится окно сохранения шаблона.
4. Введите название шаблона.
5. При необходимости введите описание шаблона.
6. Выберите каталог раздела **Экспертиза**, в который будет помещен шаблон.
7. Чтобы сохранить в шаблоне текущий заданный период, установите флажок **Сохранить временной период**.
8. Чтобы сохранить в шаблоне имеющиеся пользовательские фильтры, установите флажок **Сохранить фильтры**.
9. Нажмите на кнопку **Создать**. Система создаст шаблон поиска и отобразит соответствующее уведомление. Новый шаблон появится на панели шаблонов.

Применение шаблона

Чтобы применить шаблон:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Выберите шаблон из списка. Система применит шаблон для поиска событий.

Поиск шаблона

Чтобы найти шаблон:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Введите название требуемого шаблона в поле поиска. На панели отобразятся шаблоны, в названии которых встречается введенный запрос.

Обновление шаблона

Вы можете обновить существующий шаблон. Для этого:

1. Перейдите в раздел **Поиск**.
2. Нажмите на кнопку **Шаблоны** (). Отобразится панель шаблонов.
3. Вы можете отредактировать название и описание шаблона. Также можно установить или снять флажки **Сохранить временной период** и **Сохранить фильтры**.
4. Нажмите на кнопку **Обновить** и подтвердите обновление. Система сохранит изменения и отобразит соответствующее уведомление. В шаблоне будет сохранен поисковый запрос, введенный в поле запроса на момент редактирования.

Работа с RObject-конфигурацией шаблона поиска

- Создание шаблона поиска

- Изменение шаблона поиска
- Удаление шаблона поиска

Создание шаблона поиска

Чтобы создать шаблон поиска:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего списка опцию **Шаблон поиска**. Система отобразит окно создания шаблона.

При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля шаблона поиска, чтобы задать параметры поиска и условия фильтрации.
4. Нажмите на кнопку **Опубликовать версию**. Система создаст шаблон поиска и отобразит соответствующее уведомление. Новый шаблон появится в таблице раздела **Экспертиза**.

Чтобы шаблон поиска стал доступным для использования, его необходимо включить.

Вы можете создать черновик шаблона поиска без публикации, используя кнопку **Сохранить черновик**. Система создаст черновик и отобразит соответствующее уведомление. Черновик отобразится в таблице раздела **Экспертиза**.

Вы также можете создать конфигурацию шаблона поиска на основе существующего запроса.

Изменение шаблона поиска

Чтобы изменить шаблон поиска:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку шаблона поиска в списке. Система отобразит в правой части экрана карточку этого шаблона с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек шаблона.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента

экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.

- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

4. Внесите необходимые изменения в конфигурацию шаблона поиска.
5. Нажмите на кнопку **Обновить версию**. Система опубликует новую версию шаблона поиска с измененной конфигурацией и отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию шаблона поиска в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Система сохранит конфигурацию для дальнейшего редактирования и отобразит соответствующее уведомление. В верхней части карточки шаблона будет отображаться предупреждение о наличии неопубликованных изменений.

Удаление шаблона поиска

- Удаление доступно только для элементов экспертизы с типом создания **Пользовательский**.
- Если элемент экспертизы используется в других сущностях системы, его удаление невозможно.

Чтобы удалить шаблон поиска:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку шаблона поиска в списке. Система отобразит в правой части экрана карточку этого шаблона с подробной информацией о нем.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления шаблона.
4. Нажмите на кнопку **Удалить**. Система удалит шаблон поиска и отобразит соответствующее уведомление. Удаленный шаблон будет исключен из списка элементов экспертизы.

Вы также можете удалить группу элементов экспертизы. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные элементы и отобразит соответствующее уведомление. Удаленные элементы будет исключены из списка элементов экспертизы.

Структура шаблона поиска

Шаблон поиска содержит обязательные и опциональные поля, определяющие его логику и настройки для отбора данных.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор шаблона поиска. Должен быть уникальным в пределах экспертизы. Можно использовать UUID в качестве значения.	Строка	да
`name`	Название шаблона поиска. Допустимы латинские буквы, цифры, точки, подчеркивания и дефисы.	строка	да
`description`	Краткое описание назначения шаблона поиска.	строка	нет
`type`	Тип шаблона. Для шаблонов поиска всегда указывается `search_template`.	строка	да
`author`	Имя и контактные данные автора шаблона поиска.	строка	нет
`date`	Дата создания или изменения шаблона. Задается в формате `YYYY-MM-DD`, то есть "год-месяц-день". Поле не связано с полями **Дата создания** и **Дата изменения** в карточке элемента экспертизы.	строка	нет
`status`	Статус разработки текущей версии шаблона поиска. Может принимать любые текстовые значения. Примеры значений: • `debug` — версия в разработке; • `experimental` — экспериментальная версия; • `test` — версия для тестирования; • `stable` — стабильная версия.	строка	нет
`tags`	Список тегов, присвоенных шаблону поиска для классификации и быстрого поиска.	массив строк	нет
`version`	Версия шаблона поиска в формате Semantic Versioning.	строка	да

Поле	Описание	Тип данных	Обязательное поле
`time`	Временной интервал для выборки данных. Может включать период (`from`, `to`) или интервальное значение (`unit`, `value`).	объект	да
`rql`	RQL-запрос, определяющий логику поиска и выборки данных.	строка	да

Структура `filters`

Поле `filters` описывает настройки условий для отбора данных в шаблоне поиска. Каждый фильтр включает параметры для задания условий отбора и дополнительных параметров конфигурации.

Поле	Описание	Тип данных	Обязательное поле
`name`	Название целевого поля данных, к которому применяется фильтр.	строка	да
`operation`	Оператор сравнения, определяющий логику фильтрации. Допустимые значения: `EQ`, `NE`, `GT`, `LT`, `GE`, `LE`, `BETWEEN`, `LIKE`, `IN`.	строка	да
`values`	Массив значений, используемых для фильтрации. Для операций `IN` допускается указание нескольких значений, а для `BETWEEN` — строго два значения (границы интервала).	массив строк	да
`enabled`	Статус фильтра, указывающий, будет ли он применен (`true`) или отключен (`false`).	булево значение	да
`inverted`	Опция инверсии условия фильтрации. Если установлено в `true`, условие фильтрации будет инвертировано.	булево значение	нет
`customTitle`	Пользовательское название фильтра, отображаемое в интерфейсе на панели фильтров. Если значение не задано, используется значение поля `name`.	строка	нет

Структура `time`

Поле `time` определяет временной интервал, за который выполняется выборка данных в шаблоне поиска. Интервал может задаваться двумя способами: с помощью фиксированного периода (`from` и `to`) или через интервальные параметры (`unit` и `value`).

Поле	Описание	Тип данных	Обязательное поле
`from`	Начальная точка временного интервала в формате `YYYY-MM-DD HH:MM:SS`. Используется в паре с полем `to`.	строка	да (если `to` указано)
`to`	Конечная точка временного интервала в формате `YYYY-MM-DD HH:MM:SS`. Используется в паре с полем `from`.	строка	да (если `from` указано)
`unit`	Единица измерения временного интервала. Допустимые значения: `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH`, `YEAR`. Применяется в паре с полем `value`.	строка	да (если `value` указано)
`value`	Количество единиц, задающих продолжительность временного интервала. Например, при значениях `unit: DAY` и `value: 7` интервал составит последние 7 дней.	число	да (если `unit` указано)

Составление шаблона поиска

Для создания шаблона поиска необходимо определить ряд обязательных и опциональных полей, которые формируют структуру и логику поиска.

Поля метаданных

Заполните поля метаданных шаблона поиска:

- `id`: уникальный идентификатор шаблона поиска. Должен быть уникальным в пределах экспертизы.
- `name`: название шаблона поиска. Допустимы латинские буквы, цифры, символы подчеркивания, точки и дефисы.
- `type`: всегда указывается `search_template` для элементов экспертизы данного типа.
- `description` (опционально): краткое описание назначения шаблона.
- `author` (опционально): имя и контактные данные автора шаблона поиска.
- `version`: версия шаблона поиска согласно стандарту Semantic Versioning.
- `status` (опционально): статус разработки шаблона. Может принимать текстовые значения, такие как:
  - `draft`: черновая версия.
  - `published`: опубликованная версия.
  - `deprecated`: устаревшая версия.
- `tags` (опционально): список тегов, присвоенных шаблону для классификации и быстрого поиска.
- `date` (опционально): дата создания или изменения шаблона. Формат значения: `YYYY-MM-DD`.

Пример метаданных:

```
id: 61f0c404-5cb3-11e7-907b-000000000000
name: Failed Logins
type: search_template
description: Шаблон поиска для неудачных попыток входа
author: John Doe <johndoe@rvision.ru>
version: 1.0.0
status: published
date: 2024-01-15
tags:
  - security
  - login
```

В этом примере:

- Поле `id`: уникальный идентификатор шаблона поиска. В данном случае UUID — `61f0c404-5cb3-11e7-907b-000000000000`.
- Поле `name`: название шаблона — "Failed Logins".
- Поле `type`: тип шаблона — `search_template`.
- Поле `description`: краткое описание — "Шаблон поиска для неудачных попыток входа".
- Поле `author`: имя автора шаблона — "John Doe <johndoe@example.com>".
- Поле `version`: версия шаблона в формате `1.0.0`.
- Поле `status`: статус разработки шаблона — `published`.
- Поле `date`: дата создания шаблона — "2024-01-15".
- Поле `tags`: содержит теги `security` и `login` для классификации шаблона.

Поле `filters`

Поле `filters` используется для настройки условий фильтрации данных. Каждый фильтр определяет параметр для отбора, логический оператор и значения.

- `name`: название целевого поля данных для фильтрации.
- `operation`: логический оператор для сравнения значений (например, `EQ`, `BETWEEN`, `LIKE`, `IN`).
- `values`: массив значений для сравнения с полем.

Для операций `IN` допускается указание нескольких значений, а для `BETWEEN` — строго два значения (границы интервала).

- `inverted`: указывает, применяется ли логическое отрицание к результату фильтрации.
- `enabled`: указывает, включен ли фильтр для обработки.
- `customTitle` (опционально): заголовок фильтра, который отображается на панели фильтров.

Пример фильтра:

```
filters:
```

```
  - name: action
    values:
      - login
    operation: EQ
    inverted: false
    enabled: true
    customTitle: Action Filter
  - name: status
    values:
      - failed
    operation: EQ
    inverted: false
    enabled: true
    customTitle: Status Filter
```

В этом примере:

- Первый фильтр проверяет, равно ли значение поля `action` значению `login`.
- Второй фильтр проверяет, равно ли значение поля `status` значению `failed`.
- Оба фильтра включены для обработки (`enabled: true`), и логическое отрицание не применяется (`inverted: false`).
- В полях `customTitle` фильтрам присвоены заголовки `Action Filter` и `Status Filter`, которые отобразятся на панели фильтров.

Поле `time`

Поле `time` определяет временной интервал для выборки данных в шаблоне поиска. Может быть представлено либо фиксированным периодом, либо интервалом времени.

Фиксированный период

- `from`: начало временного интервала.
- `to`: окончание временного интервала.

Пример с фиксированным периодом:

```
time:
  from: 2024-10-25 04:00:00
  to: 2024-10-26 03:59:59
```

В этом примере данные будут отбираться за фиксированный временной период с 25 октября 2024 года, начиная с 04:00:00, по 26 октября 2024 года до 03:59:59.

Интервальный тип

- `unit`: единица измерения интервала времени (например, `DAY`, `HOUR`).
- `value`: длительность интервала в указанных единицах.

Пример с интервальным типом:

```
time:
  unit: DAY
  value: 1
```

В этом примере задается временной интервал в 1 день (`unit: DAY`, `value: 1`). Это позволяет динамически выбирать данные, например, за последние 24 часа от момента выполнения запроса.

Поле `rql`

Поле `rql` определяет запрос на языке RQL для формирования условий поиска в шаблоне.

Пример RQL-запроса:

```
rql: sourceIp = '192.0.2.1' AND dvendor = 'Linux'
```

В этом примере:

- Запрос отбирает записи, где:
  - `sourceIp`: значение поля равно `192.0.2.1`.
  - `dvendor`: значение поля равно `Linux`.

Пример шаблона поиска

Пример 94. Пример шаблона поиска "Failed Login Attempts", схема RObject

```
# Уникальный идентификатор шаблона поиска.
id: a1f0d234-6ac2-4d7e-bf93-000000000000

# Название шаблона поиска.
name: Failed Login Attempts

# Тип элемента экспертизы.
type: search_template

# Версия шаблона.
version: 1.0.0

# Описание назначения шаблона поиска.
description: Шаблон для поиска неудачных попыток входа в систему

# Автор шаблона.
author: John Doe <johndoe@example.com>

# Фильтры для настройки условий поиска.
filters:
  - name: action
```

```
    values:
      - login
    operation: EQ
    inverted: false
    enabled: true
    customTitle: Action Filter
  - name: status
    values:
      - failed
    operation: EQ
    inverted: false
    enabled: true
    customTitle: Status Filter

# Временной период для выборки данных.
time:
  unit: HOUR
  value: 1

# RQL-запрос для выборки данных.
rql: sourceIp = '192.0.2.1' AND dvendor = 'Linux'
```

Интерпретация

Метаданные шаблона поиска:

- `id`: уникальный идентификатор шаблона поиска. В данном случае UUID — `a1f0d234-6ac2-4d7e-bf93-000000000000`.
- `name`: название шаблона поиска — "Failed Login Attempts".
- `type`: тип элемента — всегда указывается как `search_template`.
- `version`: версия шаблона — `1.0.0`.
- `description`: описание назначения шаблона — "Шаблон для поиска неудачных попыток входа в систему".
- `author`: автор шаблона — "John Doe <johndoe@example.com>".

Фильтры:

- `filters`: фильтры позволяют задавать условия для отбора данных:
  - Фильтр `action` с оператором `EQ` задает условие, при котором значение поля `action` должно быть равно `login`.
  - Фильтр `status` также с оператором `EQ` отбирает события, где значение поля `status` равно `failed`.
  - Фильтры отобразятся на панели фильтров с заголовками `Action Filter` и `Status Filter`, которые заданы в полях `customTitle`.

Временной интервал:

- `time`: интервал, в пределах которого будут искаться события. В примере указан интервал в 1 час (`unit` установлен на `HOUR`, `value` равно `1`).

RQL-запрос:

- `rql`: запрос на языке RQL, определяющий логику поиска.
  - Запрос отбирает записи, где:
    - `sourceIp`: значение равно `192.0.2.1`.
    - `dvendor`: значение равно `Linux`.

Глобальные функции

В данном разделе представлен обзор структуры, принципов работы и процесса создания глобальных функций в системе. Управление глобальными функциями в системе осуществляется в разделе Экспертиза.

О глобальных функциях

Глобальная функция задает переиспользуемый блок кода на языке VRL, который можно применять при создании правил корреляции и нормализации. Такие функции позволяют настраивать фильтрацию событий, выполнять вычисления или преобразования данных.

Для использования глобальной функции в конструкторе правиле корреляции необходимо выбрать ее из списка доступных типов условий и указать значения для аргументов.

Глобальные функции помогают упростить настройку правил, избегая дублирования кода, и обеспечивают возможность многократного использования одной логики в различных сценариях.

Работа с глобальной функцией

Доступные операции над глобальной функцией:

- Создание глобальной функции
- Изменение глобальной функции
- Удаление глобальной функции

Создание глобальной функции

Чтобы создать глобальную функцию:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы, включая их статус (включен/выключен).
2. На панели инструментов нажмите на кнопку **Создать** и выберите в выпадающем списке пункт **Глобальная функция**. Система отобразит окно создания функции.

При создании элемента экспертизы поля его структуры заполняются значениями по умолчанию. Для быстрой настройки структуры в системе доступны предустановленные примеры. Чтобы заполнить поля элемента

экспертизы с помощью примера, нажмите на кнопку **Примеры** и выберите требуемый пример из выпадающего списка.

3. Заполните поля глобальной функции, чтобы определить ее логику и параметры.
4. Нажмите на кнопку **Опубликовать версию**, чтобы сохранить изменения и опубликовать созданную глобальную функцию. Система создаст глобальную функцию и отобразит соответствующее уведомление. Новая функция отобразится в таблице раздела **Экспертиза**.

Чтобы глобальная функция стала доступной для использования в условиях корреляции, ее необходимо включить.

Вы можете создать черновик глобальной функции без публикации, используя кнопку **Сохранить черновик**. Система создаст черновик и отобразит соответствующее уведомление. Черновик отобразится в таблице раздела **Экспертиза**.

Изменение глобальной функции

Чтобы изменить глобальную функцию:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы, включая их статус (включен/выключен).
2. Нажмите на строку глобальной функции в списке. В правой части экрана отобразится карточка функции с подробной информацией.
3. Выберите опцию **Изменить** в меню **Действия** (⋮) в верхней части карточки функции. Откроется окно настроек.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

4. Внесите требуемые изменения в конфигурацию функции.
5. Нажмите кнопку **Обновить версию**. Новая версия функции с измененной конфигурацией будет опубликована.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию функции в виде черновика без публикации, используя кнопку **Сохранить черновик**. Черновик будет доступен для дальнейшего редактирования, с предупреждением о наличии неопубликованных изменений.

Удаление глобальной функции

- Удаление доступно только для элементов экспертизы с типом создания **Пользовательский**.
- Если элемент экспертизы используется в других сущностях системы, его удаление невозможно.

Чтобы удалить глобальную функцию:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы, включая их статус (включен/выключен).
2. Нажмите на строку глобальной функции в списке. В правой части экрана отобразится карточка функции с подробной информацией.
3. Выберите опцию **Удалить** в меню **Действия** (⋮) в верхней части карточки. Отобразится окно подтверждения удаления.
4. Нажмите на кнопку **Удалить**. Система удалит функцию и отобразит соответствующее уведомление. Удаленная функция будет исключена из списка элементов экспертизы.

Вы также можете удалить группу элементов экспертизы. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные элементы и отобразит соответствующее уведомление. Удаленные элементы будет исключены из списка элементов экспертизы.

Структура глобальной функции

Глобальная функция содержит обязательные и опциональные поля, определяющие ее логику и параметры для обработки данных.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор глобальной функции. Должен быть уникальным в пределах экспертизы. Можно использовать UUID в качестве значения.	строка	да
`name`	Название глобальной функции. Должно начинаться с латинской буквы. В названии допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`), подчеркивания (`_`) и дефис (`-`) без пробелов.	строка	да

Поле	Описание	Тип данных	Обязательное поле
`description`	Краткое описание назначения глобальной функции.	строка	нет
`data_source`	Список источников данных, с которыми совместима глобальная функция. Используется для классификации и быстрого поиска элементов экспертизы в системе.	массив строк массив объектов	нет
`status`	Статус разработки текущей версии глобальной функции. Может принимать любые текстовые значения. Примеры значений: • `debug` — версия в разработке; • `experimental` — экспериментальная версия; • `test` — версия для тестирования; • `stable` — стабильная версия.	строка	нет
`date`	Дата создания или изменения глобальной функции. Задается в формате `YYYY-MM-DD`, то есть "год-месяц-день". Поле не связано с полями **Дата создания** и **Дата изменения** в карточке элемента экспертизы.	строка	нет
`type`	Тип объекта. Для глобальных функций всегда указывается `global_function`.	строка	да
`version`	Версия функции в формате Semantic Versioning.	строка	да
`author`	Имя и контактные данные автора функции.	строка	нет
`source`	Код глобальной функции, реализованный на языке VRL.	строка	да
`arguments`	Набор аргументов, определяющих входные данные функции. Каждый аргумент имеет тип и имя.	массив объектов	да
`fallible`	Определяет, может ли функция возвращать ошибки. По умолчанию `true`.	логический	нет
`tests`	Набор тестов для проверки корректности работы функции.	массив объектов	нет

Структура `data_source`

Поле `data_source`, представленное массивом строк, имеет структуру:

```
data_source:
- <platform> # Платформа или информационная система.
- <source> # Источник событий.
# События (EventId) или путь к журналу событий:
- <EventID_1>
- <EventID_2>
```

Поле `data_source`, представленное массивом объектов, имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`platform`	Платформа или информационная система.	строка	нет
`source`	Источник событий.	строка	нет
`events`	События (EventId) или путь к журналу событий.	массив строк	нет

Структура `arguments`

Поле	Описание	Тип данных	Обязательное поле
`keyword`	Имя аргумента, которое используется в коде функции.	строка	да
`kind`	Тип аргумента. Допустимые значения: `bytes`, `integer`, `float`, `boolean`, `object`, `array`, `timestamp`, `regex`.	строка или объект	да

Структура `tests`

Поле	Описание	Тип данных	Обязательное поле
`name`	Название теста, позволяющее идентифицировать его при проверке работы глобальной функции.	строка	да
`events`	Набор входных данных для тестирования в виде массива событий, передаваемых в функцию.	массив	да
`assertion`	Блок проверок на языке VRL, выполняемых на каждом событии из `events`. Включает `assert` и `assert_eq` для валидации ожидаемых результатов.	строка	да

Составление глобальной функции

Для создания глобальной функции необходимо определить ряд обязательных и опциональных полей, которые формируют ее поведение и логику обработки данных.

Поля метаданных

Заполните поля метаданных глобальной функции:

- `id`: уникальный идентификатор глобальной функции. Должен быть уникальным в пределах экспертизы. Можно использовать UUID в качестве значения.
- `name`: название глобальной функции. Допустимы латинские буквы, цифры и символы подчеркивания.
- `type`: всегда указывается `global_function` для элементов экспертизы данного типа.
- `description` (опционально): краткое описание назначения функции.
- `author` (опционально): имя и контактные данные автора глобальной функции.
- `version`: версия глобальной функции согласно стандарту Semantic Versioning.
- `status` (опционально): статус разработки текущей версии глобальной функции. Может принимать текстовые значения, такие как:
  - `debug`: версия в разработке.
  - `experimental`: экспериментальная версия.
  - `test`: версия для тестирования.
  - `stable`: стабильная версия.
- `date` (опционально): дата создания или изменения глобальной функции. Формат значения: `YYYY-MM-DD` (год-месяц-день). Поле не связано с полями **Дата создания** и **Дата изменения** в карточке элемента экспертизы.
- `tags` (опционально): список тегов, присвоенных глобальной функции для классификации и быстрого поиска в системе.

Пример метаданных:

```
id: a123b456-c789-0def-1112-000000000000
name: multiply_values
type: global_function
description: Функция для умножения двух чисел
author: John Doe <johndoe@example.com>
version: 1.0.0
status: stable
date: 2025-01-23
tags:
  - math
  - operations
```

В этом примере:

- Глобальная функция с уникальным идентификатором `a123b456-c789-0def-1112-000000000000` предназначена для выполнения операции умножения.
- Название функции `multiply_values` указывает на ее функциональность.
- Поле `type` имеет значение `global_function`, что обозначает тип элемента экспертизы.

- Версия функции указана как `1.0.0` в соответствии с правилами семантического версионирования.
- Поле `status` имеет значение `stable`, что указывает на завершение разработки функции.
- Поле `date` содержит дату последнего изменения или создания функции: `2025-01-23`.
- Автором функции является `John Doe`, чьи контактные данные указаны в поле `author`.
- Теги `math` и `operations` присвоены функции для облегчения поиска и классификации в системе.

Поле `source`

Поле `source` содержит основной код глобальной функции, написанный на языке VRL. Этот код реализует логику выполнения функции и использует аргументы, указанные в поле `arguments`. Код функции должен быть валидирован, а возвращаемое значение должно соответствовать ожидаемому типу, заданному в конфигурации функции.

Основные аспекты работы `source`

- **Входные данные**: аргументы, передаваемые функции, доступны внутри `source` под именами, указанными в `arguments`.
- **Логика выполнения**: последовательность операций, выполняемых над входными данными, таких как вычисления, условные конструкции и вызовы других функций.
- **Возвращаемое значение**: итоговый результат выполнения логики, указанный в `source`. Это значение возвращается функцией и должно соответствовать требованиям типов данных, заданных в конфигурации функции.

Пример реализации `source`:

```
source: !vrl |
  # Умножение значений двух аргументов.
  argA * argB
```

В этом примере:

- `argA` и `argB`: аргументы, передаваемые в функцию.
- `*`: операция умножения, результат которой возвращается как итог выполнения функции.
- Возвращаемое значение: произведение значений `argA` и `argB`.

Поддерживаемые элементы в `source`

Поле `source` поддерживает все возможности языка VRL. Подробное описание доступных выражений и функций приведено в документации по выражениям и функциям.

Основные возможности включают:

- **Операции с аргументами**: работа с переданными данными, такими как арифметические операции, манипуляции со строками и преобразования типов.
- **Условные конструкции**: использование логических выражений и операторов `if`/`else` для ветвления логики выполнения.

- **Встроенные функции VRL**: вызов встроенных VRL-функций для обработки данных (например, преобразования, фильтрации, агрегации).
- **Работа с объектами**: доступ к полям сложных объектов, переданных в функцию, через их ключи.

Примеры значений в `source`

Пример 95. Простое арифметическое выражение

```
source: !vrl |
  argA * argB
```

Выполняется умножение двух числовых аргументов `argA` и `argB`.

Пример 96. Условное ветвление

```
source: !vrl |
  if arg > 10 {
    "Value is greater than 10"
  } else {
    "Value is 10 or less"
  }
```

Функция возвращает строку в зависимости от значения аргумента `arg`.

Пример 97. Использование встроенной функции

```
source: !vrl |
  to_string(arg)
```

Функция преобразует числовое значение аргумента `arg` в строку.

Пример 98. Работа с объектом

```
source: !vrl |
  substring(arg["field_name"], 0, 5)
```

Функция извлекает подстроку из значения поля `field_name` объекта `arg`.

Поле `arguments`

Поле `arguments` задает параметры, передаваемые в глобальную функцию при ее вызове. Оно определяет структуру, тип данных и имена параметров, доступных внутри функции.

Элементы структуры `arguments`

- `keyword`: уникальное имя параметра, используемое в коде функции для ссылки на аргумент.
- `kind`: тип данных параметра.

Пример структуры `arguments`

```
arguments:
  - keyword: input_data
    kind: object
  - keyword: multiplier
    kind: integer
```

В этом примере:

- Аргумент `input_data` имеет тип `object`, что означает, что он представляет объект с произвольной структурой.
- Аргумент `multiplier` имеет тип `integer`, что указывает на ожидание целого числа.

Типы данных

Поле `kind` может принимать следующие значения типов данных:

- `bytes`: бинарные данные.
- `integer`: целые числа.
- `float`: числа с плавающей точкой.
- `boolean`: логические значения (`true` или `false`).
- `object`: объект (структура ключ-значение).
- `array`: массив.
- `timestamp`: временные метки в формате RFC 3339.
- `regex`: регулярное выражение.

Дополнительные требования

1. Имена параметров (`keyword`) должны быть уникальными в рамках одной функции.
2. Тип данных (`kind`) должен быть совместим с ожидаемыми значениями, передаваемыми при вызове функции.
3. Если аргументу передается объект, каждое поле в его структуре должно быть строго типизировано.

Поле `fallible`

Поле `fallible` в глобальной функции определяет, может ли функция возвращать ошибки.

Если в поле установлено значение `true`, функция может возвращать ошибки. При вызове такой глобальной функции нужно обработать ошибку времени выполнения путем присваивания, объединения или вызова ошибки. По умолчанию глобальные функции возвращают ошибки.

Если в поле установлено значение `false`, функция не может возвращать ошибок. При использовании такой функции не следует выполнять присваивание, объединение или вызов ошибки.

Глобальная функция может быть помечена как не возвращающая ошибок

(`fallible: false`), только если в ее коде нет вызова функций с прерыванием `!`.

Тестирование глобальной функции

Тестирование глобальной функции проверяет ее корректность и точность выполнения заданных вычислений или трансформаций на основе входных данных. Это позволяет убедиться, что функция возвращает ожидаемые результаты, корректно обрабатывает исключительные ситуации и ошибки, а также стабильно работает в продуктивной среде.

Если глобальная функция взаимодействует с внешними данными (например, активными списками или таблицами обогащения), рекомендуется настроить искусственные данные через поле `setup_tests`. Это позволяет моделировать работу функции в условиях, близких к продуктивным, без обращения к реальной инфраструктуре.

Поле `tests`

Поле `tests` определяет набор тестов для проверки работы функции. Каждый тест содержит описание входных данных, условий проверки, и ожидаемого результата.

Элементы структуры `tests`:

- `name`: название теста для его идентификации.
- `events`: список входных данных, которые подаются на вход функции в процессе тестирования. Входные данные состоят из следующих обязательных полей:
  - `$arguments` — список передаваемых аргументов.
  - `$event` — массив тестовых событий. Эти события моделируют данные, которые передаются функции при ее реальной работе.
- `assertion`: выражение или набор выражений на языке VRL, проверяющих корректность результата выполнения функции.

Пример структуры `tests`

```
tests:
  - name: Тест умножения чисел
    events:
      - { "$arguments": { "argA": 50, "argB": 10 }, "$event": {} }
    assertion: !vrl |
      assert_eq!(.[0], 50, "Ожидается произведение 50")
```

В этом примере:

- `name`: тест называется "Тест умножения чисел".
- `events`: входное событие включает два аргумента, `argA` и `argB`.
- `assertion`: проверяется, что вызов функции возвращает значение `50`.

Тестовые события

Массив `events` содержит данные, передаваемые в функцию в процессе тестирования. Каждый элемент массива представляет собой объект, где ключи соответствуют именам аргументов функции, а значения — типам данных, указанным в `arguments`.

Для работы с временными метками (например, типом `TIMESTAMP`) значения в событиях должны быть указаны в формате `t'<значение>'`. Например: `t'2024-05-15 06:28:15'`. Это гарантирует корректную обработку времени.

Условия проверки

Поле `assertion` задает выражения на языке VRL, которые проверяют результат работы функции. Условия могут включать проверки значений, использование логических операторов, или проверку обработки исключительных ситуаций.

Создание и выполнение тестов

Для проверки глобальной функции создается набор тестов, каждый из которых включает следующие элементы:

1. Описание теста (`name`): краткое описание цели и ожидаемого результата теста.
2. События (`events`): список входных данных, которые подаются на вход функции в процессе тестирования. Для входных данных необходимо заполнить следующие поля:
   - `$arguments` — список передаваемых аргументов.
   - `$event` — массив тестовых событий.
3. Утверждения (`assertions`): логика проверки результата работы функции на основе переданных событий и условий.

Пример:

```
tests:
  - name: Проверка функции деления
    events:
      - { $arguments: { dividend: 10, divisor: 2 }, $event: {}}
    assertion: !vrl |
        assert_eq!(.[0], 5, "Ожидается результат 5")
```

Логика проверки (`assertions`) позволяет убедиться, что возвращаемый функцией результат соответствует ожидаемому. Если утверждение выполняется, это подтверждает корректность работы функции. Если же результат отличается от ожидаемого, это может указывать на необходимость доработки функции.

- Перед запуском тестов убедитесь, что значения в `events` соответствуют формату и типам, ожидаемым функцией.
- Используйте результаты тестов для оптимизации логики и повышения устойчивости функции.

Поле `setup_tests`

Для тестирования элементов экспертизы, которые используют внешние зависимости, то есть взаимодействуют с активными списками, таблицами обогащения или глобальными функциями, поле `setup_tests` является обязательным.

Поле `setup_tests` в конфигурации предназначено для определения тестового окружения, включая предварительную настройку искусственных данных для активных списков, таблиц обогащения и глобальных функций, используемых в элементе экспертизы. Это обеспечивает проверку правила на корректность выполнения и взаимодействие с внешними данными до его применения в операционной среде.

Поле `setup_tests` используется для:

- Имитации взаимодействия с активными списками, таблицами обогащения и глобальными функциями.
- Проверки логики правила или глобальной функции с учетом ожидаемых данных обогащения.

Процедура настройки `setup_tests`:

1. В корне конфигурации укажите секцию `setup_tests`.
2. Внутри секции `setup_tests` через ключ `enrichment_tables`, `active_lists` или `global_functions` задайте искусственные таблицы обогащения, активные списки и глобальные функции, используемые правилом.
3. Для каждого искусственного элемента укажите его структуру и набор искусственных данных, которые будут использоваться при тестировании.

   Искусственные данные должны соответствовать ожидаемым данным из реальных таблиц обогащения, активных списков или глобальных функций и позволять полноценно проверить работу элемента экспертизы.

Пример 99. Структура `setup_tests` для активного списка:

```
setup_tests:
  active_lists:
    <имя_списка>:
      <поле>:
        - [<значение_поля1>]
        - [<значение_поля2>]
```

Пример 100. Структура `setup_tests` для таблицы обогащения:

```
setup_tests:
  enrichment_tables:
    <имя_таблицы>:
      fields:
        - <поле1>
        - <поле2>
```

```
      data:
        - [<значение1_поля1>, <значение1_поля2>]
        - [<значение2_поля1>, <значение2_поля2>]
```

Пример 101. Структура `setup_tests` для глобальной функции:

```
setup_tests:
  global_functions:
    <имя_функции>:
      arguments:
        - keyword: <имя_параметра1>
          kind: <тип_параметра1>
        - keyword: <имя_параметра2>
          kind: <тип_параметра2>
      values_map:
        - argument: <аргументы1>
          event: <событие1>
          return: <результат1>
      default: <результат_по_умолчанию>
      fallible: <логическое значение>
```

При вызове искусственная глобальная функция возвращает значение, заданное в `values_map`, если найдена запись с совпадающими аргументами. Если подходящей записи нет, возвращается значение из поля `default`.

Пример глобальной функции

Пример 102. Пример глобальной функции "Multiplication Function", схема RObject

```
# Универсальный уникальный идентификатор функции.
id: d2f0c404-5cb3-11e7-907b-000000000000

# Название глобальной функции.
# Должно быть уникальным в пределах коллектора.
name: multiplication_example

# Тип элемента экспертизы. Всегда имеет значение `global_function` для глобальных
функций.
type: global_function

# Версия функции.
version: 1.0.0

# Краткое описание назначения глобальной функции.
description: Функция умножения двух чисел

# Имя и контактные данные автора глобальной функции.
author: John Doe <johndoe@example.com>

# Статус разработки функции.
```

```
# Возможные варианты: `debug`, `experimental`, `test`, `stable`.
status: stable

# Дата создания или последнего изменения функции, формат YYYY-MM-DD.
date: 2025-01-23

# Категории, по которым функции группируются.
tags:
  - math
  - example

# Исполняемый код функции на языке VRL.
source: !vrl |
  # Функция возвращает произведение двух аргументов.
  if is_integer(argA) && (is_integer(argB)) {
    argA * argB
  } else {
    0
  }

# Параметры функции.
arguments:
  # Имя первого аргумента.
  - keyword: argA
    # Тип первого аргумента.
    kind: integer
  # Имя второго аргумента.
  - keyword: argB
    # Тип второго аргумента.
    kind: integer

# Опциональное. Определяет, может ли функция возвращать ошибки. По умолчанию 'true'.
fallible: true

# Список тестов функции.
tests:
  # Наименование теста, уникальное для каждого теста.
  - name: Multiplication Test 1
    # Тестовые данные.
    events:
      - { $arguments: { argA: 10, argB: 2 }, $event: {}}
    # Проверка для тестовых данных.
    assertion: !vrl |
      assert_eq!(.[0], 20, "Ожидается произведение 20")

  # Наименование теста для проверки обработки ошибок.
  - name: Multiplication Test Invalid Data
    # Тестовые данные с некорректным форматом.
    events:
      - { $arguments: { argA: "invalid", argB: 2 }, $event: {}}
    # Проверка для тестовых данных.
```

```
    assertion: !vrl |
      assert_eq!(.[0], 0, "При некорректном вводе ожидается результат 0")
```

Интерпретация

Метаданные глобальной функции:

- `id`: уникальный идентификатор функции — `d2f0c404-5cb3-11e7-907b-000000000000`.
- `name`: название функции — `multiplication_example`.
- `type`: тип объекта экспертизы — всегда `global_function` для глобальных функций.
- `version`: версия функции — `1.0.0`.
- `description`: описание назначения функции — "Функция умножения двух чисел".
- `author`: автор функции — `John Doe <johndoe@example.com>`.
- `status`: статус разработки функции — `stable`, что означает завершенную разработку и готовность к использованию.
- `date`: дата создания или последнего изменения функции — `2025-01-23`.
- `tags`: категории функции, используемые для классификации — `math` и `example`.

Код функции:

- `source`: основной код функции, написанный на языке VRL, реализует умножение значений аргументов `argA` и `argB` при условии, что в оба аргумента переданы целые числа. В противном случае в качестве результата функции возвращается значение `0`.

Параметры функции:

- `arguments`: список параметров, передаваемых функции.
- `keyword`: имя аргумента. В данном случае это `argA` и `argB`.
- `kind`: тип данных аргумента. В данном случае это `integer` (целое число).

Тесты:

- `tests`: набор тестов для проверки правильности выполнения функции.
- Тест 1:
  - `name`: название теста — "Multiplication Test 1".
  - `events`: данные для теста содержат два значения: `argA = 10` и `argB = 2`.
  - `assertion`: проверяется, что функция `multiplication_example` вернет `20` при переданных аргументах.
  - Сообщение об ошибке, возвращаемое в случае несоответствия: `"Ожидается произведение 20"`.
- Тест 2:
  - `name`: название теста — "Multiplication Test Invalid Data".
  - `events`: данные содержат некорректное значение `argA = "invalid"` и корректное `argB = 2`.
  - `assertion`: проверяется, что функция обрабатывает невалидные данные и возвращает `0`, если

вход некорректен.

- Сообщение об ошибке, возвращаемое в случае несоответствия: "При некорректном вводе ожидается результат 0".

Схемы доменов

В данном разделе представлен обзор структуры, процесса создания и валидации схем доменов в системе. Управление схемами доменов в системе осуществляется в разделе **Экспертиза**.

О схемах доменов

Схема домена — главный компонент домена, содержащий все его необходимые настройки. С помощью схемы можно описать конфигурацию интерфейса домена и определить просмотр сущностей домена. Конфигурация определяет набор элементов в интерфейсе и их отображение в окне системы.

Работа со схемой домена

Доступные операции над схемой домена:

- Создание схемы домена
- Изменение схемы домена
- Удаление схемы домена

Создание схемы домена

Чтобы создать схему домена:

1. Перейдите в раздел **Экспертиза**. Система отобразит список элементов экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего списка опцию **Схема домена**. Система отобразит окно создания схемы домена.
3. Заполните поля схемы домена, чтобы определить логику ее работы.
4. Нажмите на кнопку **Запустить тесты**, чтобы провести валидацию настроек добавленной схемы. Результаты валидации отобразятся в поле **Результат**. Поле можно скрыть, нажав на кнопку **Свернуть**.

   Если при валидации схемы домена будут найдены ошибки, исправьте их, используя список ошибок и пояснения к ним.

5. Нажмите на кнопку **Опубликовать версию**, чтобы сохранить изменения и опубликовать добавленную схему домена. Система отобразит соответствующее уведомление. Новая схема домена появится в таблице раздела **Экспертиза**.

   Чтобы схема домена стала доступной для создания домена, ее необходимо включить.

   Вы можете создать схему домена без ее публикации (в виде черновика) с помощью кнопки **Сохранить черновик**. Система отобразит соответствующее уведомление. Черновик появится в

таблице раздела **Экспертиза**.

Изменение схемы домена

Чтобы изменить схему домена:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку схемы домена в списке. Система отобразит в правой части экрана карточку с подробной информацией о схеме.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек схемы.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

4. Внесите требуемые изменения в конфигурацию схемы.
5. Нажмите на кнопку **Обновить версию**. Новая версия схемы с измененной конфигурацией будет опубликована. Система отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию схемы в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки схемы будет отображаться предупреждение о наличии неопубликованных изменений.

Удаление схемы домена

Чтобы удалить схему домена:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку схемы домена в списке. Система отобразит в правой части экрана карточку с подробной информацией о схеме.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления схемы.
4. Нажмите на кнопку **Удалить**. Система удалит элемент и отобразит соответствующее уведомление. Удаленная схема домена будет исключена из списка элементов экспертизы.

Обработка ошибок валидации схемы домена

При добавлении схемы домена могут возникать различные ошибки. Ниже приведен список возможных ошибок с описаниями.

Ошибка	Описание
`[ERROR] YamlParseError: missing required field () at line () column ()`	Отсутствуют все обязательные поля схемы RObject или схемы домена. Убедитесь, что все необходимые поля указаны.
`[ERROR] YamlParseError: Incorrect structure or line shift: could not find expected () at line () column ()`	Структура RObject некорректна. Проверьте правильность структуры.
`[ERROR] UnreachableReference: the reference to entity () is unreachable at line () column ()`	Ссылка на сущность в схеме текущего домена недостижима. Проверьте корректность ссылки на сущность.
`[ERROR] UnreachableReference: the reference to attribute is unreachable () at line () column ()`	Ссылка на атрибут в схеме текущего домена недостижима. Проверьте корректность ссылки на атрибут.
`[ERROR] InvalidDefinition: the constraint is not supported by attribute type at line () column ()`	Атрибут имеет ограничение, несовместимое с типом данных. Убедитесь, что все ограничения и параметры предварительно настроенных типов данных соответствуют схеме.
`[ERROR] UnreachableReference: the entity or attribute at another domain () is unreachable: () at line () column ()`	Сущность или атрибут домена, на который ссылается текущий домен, не зарегистрированы. Убедитесь, что все сущности и атрибуты других доменов, на которые ссылается текущий домен, зарегистрированы в текущей инсталляции.
`[ERROR] NotCompatible: the entity at the new version of domain scheme is changed or deleted: () at line () column ()`	В новой версии схемы изменены или удалены сущности. Убедитесь, что в новой версии схемы домена не было изменений или удалений сущностей. Атрибуты сущностей разрешено изменять или удалять.
`[ERROR] UnreachableReference: the entity or attribute at the current domain scheme is unreachable to domain (): () at line () column ()`	Сущность или атрибут текущего домена, на который ссылается другой домен, недостижимы. Убедитесь, что все сущности и атрибуты текущей схемы, на которые ссылаются другие домены, достижимы.
`[ERROR] NotCompatible: the attribute type at the current domain scheme is not compatible at line () column ()`	Несовместимая схема: тип атрибута несовместим. Убедитесь, что схема обратно совместима со связанными доменами.
`[ERROR] NotCompatible: the attribute constraint at the current domain scheme is not compatible at line () column ()`	Несовместимая схема: ограничение атрибута несовместимо. Убедитесь, что схема обратно совместима со связанными доменами.

Структура схемы домена

Ознакомьтесь с правилами нотации в схеме домена. Это позволит присваивать корректные названия создаваемым элементам схемы. Эти названия будут отображаться в интерфейсе системы.

Структура метаданных

Запись метаданных включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Уникальный идентификатор схемы.	строка	Полное имя (FQDN) домена, который описывается схемой.
`type`	да		Тип элемента экспертизы.	перечисление	Всегда устанавливается значение `domain`.
`version`	да		Версия схемы домена в формате Semantic Versioning.	строка	
`name`	да		Название схемы домена.	строка	Допустимы строчные и прописные буквы латинского и кириллического алфавитов (`a–z`, `A–Z`, `а–я`, `А–Я`), цифры (`0–9`), точки (`.`), подчеркивания (`_`) и дефис (`-`) без пробелов.
`tags`	нет	Пустой массив	Теги для классификации и поиска схемы в системе.	массив строк	
`description`	нет	Пустая строка	Описание схемы домена.	строка	

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`author`	нет	Пустая строка	Имя и контактные данные автора.	строка	
`tags`	нет		Теги для классификации и поиска схемы в системе.	строка	

Структура `entities`

Поле `entities` представляет собой массив записей сущностей, входящих в состав домена. В запись сущности необходимо включить поле `attributes`, содержащее описание массива атрибутов сущности.

Запись сущности включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`id`	да		Уникальный идентификатор сущности.	Data ID	Подставляется вместо поля `label`, если не определены поля `label` и `name`.
`name`	нет	Пустая строка	Название сущности. Подставляется вместо поля `label`, если оно не определено.	строка	

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`label`	нет	Если не указано, используются идентификаторы объекта, домена и сущности. См. раздел о работе с полем.	Комбинация атрибутов сущности для отображения в интерфейсе (лейбл).	шаблон	Поиск в домене осуществляется по атрибутам сущности, ссылки на которые добавлены в этом поле. В шаблоне нельзя ссылаться на поле `label` и атрибуты другого объекта через ссылку на объект. См. раздел о работе с полем.
`description`	нет	Пустая строка	Описание сущности.	строка	Не выводится в представлениях. Отображается на визуализации схемы домена.
`kind`	нет	`regular`	Вид сущности.	перечисление	В текущей реализации системой всегда обрабатываются сущности с видом `regular` — стандартная сущность.
`attributes`	нет	Пустой массив	Описание атрибутов сущности.	массив	См. описание поля `attributes`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`sorting`	нет	Пустой массив	Описание правил сортировки, которые определяют наборы атрибутов, по которым будет осуществлена сортировка списка объектов.	массив	См. описание поля `sorting`.
`identification`	нет	Пустой массив	Описание правил идентификации объектов, поступающих в систему.	массив	См. раздел о работе с полем.
`reconciliation`	нет	Пустой массив	Описание правил реконсиляции объектов, поступающих в систему.	массив	См. раздел о работе с полем.
`projection`	нет	Пустой массив	Проекция, позволяющая посредством API в одном запросе получить атрибуты для отображения данных другой сущности.	массив	См. описание поля `projection`.
`searching`	нет	Пустой массив	Описание набора атрибутов, по которым будет осуществлен поиск в списке объектов.	массив	См. описание поля `searching`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`inherits`	нет	Пустая строка	Ссылка на сущность, которая является родителем для текущей сущности.	Data ID Ref	См. описание поля `inherits`.

▼ **Схема**

Вариант схемы записи, согласно которой формируется описание сущности в схеме:

```
entities:
 # Уникальный идентификатор сущности.
 - id:
   # Название сущности.
   name:
   # Описание лейбла сущности для представлений.
   label:
   # Описание сущности.
   description:
   # Перечисление элементов массива атрибутов для сущности.
   attributes:
```

Структура `label`

Поле `label` задается в схеме домена опционально, через указание шаблона. Если поле задано, то в представлениях сущности система будет использовать в качестве отображаемых названий объектов значения атрибута или атрибутов объектов, указанных в шаблоне.

Поле `label` со ссылкой на один атрибут задается в следующем формате: `label: "{<attribute>}"`. Здесь `<attribute>` — ссылка на атрибут сущности, позволяющий идентифицировать объект в представлении.

Если атрибут, на который вы ссылаетесь в поле `label`, недостаточно информативен для однозначной идентификации объекта, в поле `label` можно сослаться на несколько атрибутов. Также в это поле можно добавлять поясняющие комментарии.

Если поле `label` не задано, то в представлении для отображения названия объекта используются идентификаторы объекта, домена и сущности: `{id} ({domainId}/{entityId})`.

▼ **Пример**

В домене **assets.example.com** задана сущность Оборудования с идентификатором **asset**. Для сущности Оборудования не задано поле `label`.

В представлении объекта с UUID, равным **ajdmbfdd**, вместо названия объекта Оборудования будет отображаться следующая связка: **ajdmbfdd (assets.example.com/asset)**

Рекомендуется задавать поле `label` в описании сущности, чтобы облегчить пользователю работу с представлениями.

Структура `attributes` (сущность)

Поле `attributes` представляет собой массив записей атрибутов определенной сущности.

Запись атрибута в массиве описывается следующими полями:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`id`	да		Уникальный идентификатор атрибута	Data ID	
`name`	нет	Пустая строка	Название атрибута	строка	
`description`	нет	Пустая строка	Описание атрибута	строка	Не выводится в представлениях. Отображается в схеме домена.
`dataType`	да		Тип данных атрибута	перечисление, Data ID Ref или FQID	Можно указывать как встроенный тип данных, так и идентификатор пользовательского типа данных. Если пользовательский тип данных определен в другом домене, нужно указывать в этом поле FQID пользовательского типа данных.
`entity`	да (для `Reference`)		Ссылка на сущность	Data ID Ref или FQID	Поле доступно только для типа данных `Reference`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`constraints`		Пустой массив	Валидация атрибута	массив	Для каждого типа данных задается свой набор ограничителей с набором полей. См. описание поля `constraints`.
`readonly`	нет	`false`	Возможность редактирования атрибута	булево	Доступные значения: • `true` — значение атрибута задается при его создании. В дальнейшем это значение обновлять нельзя. Пример: ID из внешней системы, который задается при синхронизации и больше не обновляется. • `false` — значение атрибута можно обновлять.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`calculation`	да (для `Calculation`)		Ссылка на расчет, результат которого будет являться значением атрибута	объект	Поле доступно только для типа данных `Calculation`.

▼ **Схема описания атрибута**

```
attributes:
  # Уникальный идентификатор атрибута.
  - id:
    # Название атрибута.
    name:
    # Описание атрибута.
    description:
    # Тип данных атрибута.
    dataType:
    # Массив описаний валидации атрибута.
    constraints:
      # Первое описание валидации.
      - kind:
      # Второе описание валидации.
      - kind:
```

Структура `constraints`

В поле `constraints` задаются виды валидации атрибута.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`kind`	да		Описание валидации	массив	См. список доступных значений.
`message`	нет	Пустая строка	Сообщение об ошибке	строка	Сообщение об ошибке при срабатывании ограничителя на входящих данных.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`min`	нет	`0`	Минимальное возможное количество символов в атрибуте	целое число	Поле применимо только для следующих видов ограничителей: • `range`; • `length`.
`max`	нет	2^{31}-1 (для числа символов) 4 МБ (для размера файла)	• Максимальное возможное количество символов в атрибуте. • Максимальный размер файла.	целое число	Поле применимо только для следующих видов ограничителей: • `range`; • `length`; • `size`.
`regexp`	нет		Регулярное выражение	строка	Поле применимо только для ограничителя `regexp`.
`application`	нет		Допустимые типы файлов	строка	Поле применимо только для ограничителя `mime_types`.

Доступные значения в поле `kind`

Доступны следующие типы валидации:

- `required` — обязательный атрибут, то есть значение атрибута не должно быть пустым.
- `range` — ограничения числового значения (используется только для числовых типов данных: `Integer`, `Decimal`, `Float`).
- `length` — ограничение количества символов в атрибуте. Необходимо указать максимальное и/или минимальное количество символов в полях `max` и `min`.
- `regexp` — задание текстового шаблона в формате регулярного выражения. Необходимо задать выражение в поле `regexp`.
- `size` — ограничение размера файла. Необходимо указать минимальный и/или максимальный объем файла в полях `min` и `max`.

- `mime_types` — ограничение типов для загрузки файлов в это поле. Необходимо указать допустимые типы файлов.

Структура `calculation`

В поле `calculation` указывается ссылка на расчет, результат которого будет являться значением атрибута.

Поле	Обязательное поле	Описание	Тип данных
`schemaId`	да	Идентификатор схемы, в которой определен требуемый расчет	строка
`id`	да	Идентификатор расчета	строка

Пример 103. Пример описания поля `calculation` для сущности Требование (`requirement`)

```
entities:
  - id: Requirement
    attributes:
      - id: assessmentIndex
        name: Результат расчета индекса соответствия
        dataType: Calculation
        calculation:
          schemaId: evo.compliance.audit.calculations
          id: audit_assessment_compliance_index
```

Правила сортировки (`sorting`)

Поле `sorting` представляет собой массив записей, описывающих правила сортировки атрибутов сущности.

Запись правила сортировки включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`id`	да		Уникальный идентификатор правила сортировки из описания сущности	Data ID	
`name`			Название правила сортировки	строка	

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`by`	да		Сортируемые атрибуты, перечисленные в последовательности сортировки	массив	См. описание поля `by`.
`createIndex`	нет		Параметр, определяющий, будут ли индексироваться перечисленные атрибуты	булево	• `true` — атрибуты индексируются. • `false` — атрибуты не индексируются.

Структура `by`

Поле `by` представляет собой запись правила сортировки.

Запись правила сортировки включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`attribute`	да		Ссылка на атрибут, по которому производится сортировка.	Data ID Ref	Если в правиле указана ссылка на атрибут с типом данных `Reference`, система подставляет при сортировке объектов вместо значения самого атрибута значение поля `label` объекта, на который указывает ссылка.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`direction`	да	`asc`	Порядок сортировки	перечисление	Доступные значения: • `asc` — по возрастанию. • `desc` — по убыванию.

Структура `identification`

В схеме домена можно опционально задавать правила идентификации и реконсиляции. Если правило отсутствует, каждый полученный объект считается уникальным и записывается в базу данных отдельно.

Система обрабатывает входящие данные пакетами. Правила работают с уже нормализованными данными. После того как вы измените параметры правил и в систему поступят новые данные, измененные правила будут применяться как к данным, поступившим в систему после сохранения изменений, так и к данным, уже хранившимся в базе данных.

Правила идентификации содержат ссылки на атрибут или набор атрибутов сущности. По этим атрибутам данные об объекте, получаемые из источников, проверяются на уникальность.

Алгоритм работы правила:

1. Система ищет дубликаты объектов во входящих данных и отфильтровывает их.
2. После отработки фильтрации дубликатов система сравнивает объекты с объектами, хранящимися в БД.
3. Если система находит в БД дубликаты объекта, они будут удалены. Остается по одному экземпляру объекта на стороне входящих данных и на стороне БД.
4. Система записывает данные о полученных объектах в БД

По итогам идентификации система выявляет дубликаты объектов, который либо отфильтровываются, либо агрегируются — в зависимости от правил реконсиляции.

Логика формирования правил идентификации:

1. В правиле можно связывать атрибуты логическими операторами AND/OR, а также задавать вложенные условия. Допускается не более трех уровней вложенности.
2. На одном уровне вложенности можно одновременно использовать разные логические операторы.
3. Если в блоке задано несколько правил, между ними работает логика OR.
4. Правила располагаются в порядке убывания приоритета.

При обработке входящих данных можно дополнительно использовать фильтрацию данных с помощью правила на языке преобразования данных VRL (Vector Remap Language). VRL-правило

фильтрации описывается в рамках правила идентификации в поле `filter`.

VRL-правило позволяет выполнить следующие действия:

1. Первичная дедупликация входящих данных. Данные сначала фильтруются согласно VRL-правилу, а затем дедуплицируются по атрибутам, заданным в правиле. Все найденные дубликаты перезаписываются объектом, который был идентифицирован первым во входящих данных.
2. Фильтрация данных, полученных из БД. Данные собираются на основе атрибутов, заданных в правиле, а затем фильтруются с помощью VRL-правила.

Поле `identification` представляет собой массив правил идентификации.

Запись включает следующие поля:

Поле	Описание	Тип данных	Комментарий
`id`	Уникальный идентификатор правила (обязательное поле).	строка	
`rule`	Описание правила.	массив	См. описание поля `rule`.
`filter`	Фильтрация входящих данных с помощью VRL-правила.	массив	Поле содержит команду на языке VRL, возвращающую ответ в булевом формате.

Структура `rule`

Поле содержит массив ссылок на атрибуты, по которым происходит идентификация в рамках правила. Можно использовать до трех уровней вложений.

Запись включает следующие поля:

Поле	Обязательное поле	Описание	Тип данных	Комментарий
`attribute`	да	Ссылка на атрибут, по которому происходит идентификация.	строка	
`and`	нет	Логический оператор И.	строка	Поле содержит массив ссылок на атрибуты, которые учитываются в правиле по логике И. Может использоваться на разных уровнях в правиле.

Поле	Обязательное поле	Описание	Тип данных	Комментарий
`or`	нет	Логический оператор ИЛИ.	строка	Поле содержит массив ссылок на атрибуты, которые учитываются в правиле по логике ИЛИ. Может использоваться на разных уровнях в правиле.

Структура `reconciliation`

Поле `reconciliation` представляет собой массив правил реконсиляции. Правила реконсиляции определяют набор атрибутов объектов, подчиняющихся логике агрегации, и набор источников, данные от которых будут учитываться при перезаписи значений атрибутов.

Если правило отсутствует, по умолчанию система будет перезаписывать значения атрибутов по мере поступления от источников в порядке очередности.

Для сущности нельзя определить правило реконсиляции, если не задано правило идентификации.

Запись включает следующие поля:

Поле	Обязательное поле	Описание	Тип данных	Комментарий
`attribute`	нет	Ссылка на атрибут и список приоритетных источников для этого атрибута	строка	См. описание поля `attribute`.
`default`	да	Массив записей источников для всех атрибутов сущности по умолчанию	массив	См. описание поля `default`.

Логика задания источников в правилах реконсиляции

Для каждого атрибута в правиле задается список источников, которые считаются наиболее приоритетными при сборе данных по атрибуту. Идентификаторов источников в списке располагаются в порядке убывания приоритета.

Значение параметра "*" позволяет учитывать в правиле все источники, не указанные в правиле. Они будут иметь одинаково низкий приоритет по сравнению с указанными источниками.

Всегда указывайте значение параметра "*" в правиле в кавычках.

Если значение параметра "*" не задано в правиле, данные для реконсиляции берутся только из указанного списка источников.

Для всех атрибутов, не указанных в правиле, нужно создать единое правило приоритетов с помощью параметра `default` . Если нужно определить единую логику приоритизации источников для всех атрибутов сущности, задайте параметр `default` без указания конкретных атрибутов.

Логика перезаписи значений атрибутов

Перезапись значений происходит после срабатывания правила идентификации. При этом система учитывает заданные правила реконсиляции. Если правила реконсиляции не задавались, используется логика, заданная в поле `default`.

- Данные из разных источников перезаписываются с учетом определенной в правиле приоритизации.
- При обработке данных, поступивших в рамках одного запроса, учитываются самые последние полученные значения.
- Если текущее значение не равно `null` и приоритет входящих данных не ниже приоритета текущих данных, данные перезаписываются.
- Значение не перезаписывается на `null` ни при каких условиях.

Структура `attribute`

Поле `attribute` содержит ссылку на атрибут и массив записей источников, считающихся приоритетными для этого атрибута.

Запись включает следующие поля:

Поле	Обязательное поле	Описание	Тип данных	Комментарий
`sourcePriority`	да	Название приоритетного источника для атрибута	массив	Поле содержит массив значений поля `source`, где перечисляются приоритетные источники для атрибута. Поле может принимать значения: • название источника; • звездочка (`"*"`). В этом случае все источники, не указанные в списке, также учитываются при реконсиляции, но имеют одинаково низкий приоритет по сравнению с указанными источниками.

Структура `default`

Поле `default` содержит массив записей источников, считающихся приоритетными для всех атрибутов сущности, не указанных отдельно.

Запись включает следующие поля:

Поле	Обязательное поле	Описание	Тип данных	Комментарий
`source`	да	Название источника	строка	Поле может принимать значения: • название источника; • звездочка (`"*"`). В этом случае все источники, не указанные в списке, также учитываются при реконсиляции, но имеют одинаково низкий приоритет по сравнению с указанными источниками.

Логика обработки пользовательских данных

Пользовательские данные обрабатываются точно также, как и данные, поступающие из других источников. После создания или редактирования объекта пользователем через UI сохраненные пользовательские данные будут обработаны правилами идентификации и реконсиляции. При настройке приоритетов источников в правиле реконсиляции можно назначить пользовательским данным определенный приоритет. Для этого нужно указать в качестве ID интеграции с источником значение users.

Структура `projection`

Проекция используется, если при определении атрибута текущей сущности необходимо сослаться в этом атрибуте на исходную сущность и отображать ее данные в представлении. Для автоматического подбора дополнительных данных объекта исходной сущности при запросе значения атрибута со ссылкой, в исходной сущности указывается проекция с перечислением всех необходимых атрибутов. Проекция содержит массив записей избранных атрибутов исходной сущности и задается в поле `projection`.

Запись проекции описывается следующими полями:

Поле	Обязательное поле	Описание	Тип данных
`attribute`	да	Ссылка на атрибут	Data ID Ref

Структура `searching`

Если в поле `searching` задано правило поиска, система будет производить поиск не только по лейблу, но и по указанным в правиле атрибутам связанной сущности.

Чтобы система производила поиск по связи, необходимо в самой связи в поле `searchable` установить значение `true`.

Пример 104. Пример

```
entities:
 - id:
   name: Device
   # Поле, в котором задается правило поиска.
   searching:
      # Список атрибутов сущности, по которым производится поиск.
      attributes:
         - objectGuid
         - commonName
```

Структура `inherits`

Наследование позволяет использовать описанные свойства сущностей при работе с другими сущностями. Оно задается при помощи поля `inherits` в описании сущности. Для сущности можно указать ссылку только на одного родителя.

Каждая новая сущность-потомок будет добавлять собственную структуру к наследуемой структуре и не влиять на структуру родителей при обновлении домена. При запросе данных объектов потомка данные родителей будут включаться в вывод данных, если это определено в схеме. К наследуемым свойствам относятся лейбл (человекочитаемый отображаемый ID), атрибуты и действия.

Пример 105. Пример для родителя и потомка, находящихся в одном домене

```
entities:
  - id: Device
    name: Оборудование
    description: Оборудование
  - id: Server
    name: Сервер
    description: Сервер
    # В этом поле задается наследование: сущность Сервер является потомком сущности
Оборудование.
    inherits: Device
```

Родитель и потомок могут находиться в одном или в разных доменах. Это нужно учитывать при задании ссылки на родителя.

Местонахождение сущностей	Формат ссылки на родителя
Сущности находятся в одном домене.	Data ID Ref (ссылка на ID сущности в текущей схеме).
Сущности находятся в разных доменах.	FQID (полный доменный идентификатор).

Поле `label` (лейбл) является дополнительным неуникальным, но дружелюбным к пользователю идентификатором объекта. Если лейбл не задан явно для потомка, в представлениях для потомка наследуется лейбл родителя.

В поле `attributes` в описании потомка включаются только его собственные атрибуты. Ссылки на атрибуты родителя в описание не включаются.

При описании представлений и действий потомка ссылки на атрибуты сущностей родителя и потомка нужно формировать, учитывая следующие правила:

- На атрибут потомка можно ссылаться кратко, указывая только ID атрибута. Формат: **ID атрибута**.
- На атрибут родителя в этом же домене нужно ссылаться одним из следующих способов:
  - указывать только ID сущности и ID атрибута. Формат: **ID сущности.ID атрибута**;
  - указывать FQDN домена, ID сущности и ID атрибута. Формат: **FQDN/ID сущности.ID атрибута**.

На атрибут родителя в другом домене нужно ссылаться, упоминая FQDN домена, ID сущности и ID атрибута. Формат: **FQDN/ID сущности.ID атрибута**.

Ссылка на атрибут указывается от той сущности, в которой он изначально объявлен. Таким образом, полная ссылка на атрибут в формате **FQDN/ID сущности.ID атрибута** может использоваться во всех случаях.

Правила сортировки, заданные для родителя, не наследуются потомком. Для описания сортировки в потомке нужно создавать собственное правило сортировки. Вы можете указать в правиле сортировки атрибуты родителя по FQID. Правила задания ссылок на атрибуты при создании правил сортировки аналогичны правилам ссылок при описании представлений и действий.

Проекция, заданная для родителя, не наследуется потомком. Правила задания ссылок на атрибуты при создании проекций аналогичны правилам ссылок при описании представлений и действий.

Наследование действий

Если в действии объявлена сущность родителя, это действие может быть вызвано для его потомков. Для этого необходимо описать в представлении потомка виджет со ссылкой на это действие родителя. Действие создания объекта достаточно описать только для родителя. Если вы будете ссылаться в описании потомка на виджет родителя, которые использует это действие, оно станет доступно в интерфейсе потомка.

Работа с интерфейсами объекта и списка объектов

Рекомендуемый подход при создании схемы домена с использованием наследования:

1. Опишите все необходимые интерфейсы для сущности родителя, то есть определите для него все требуемые панели, модальные окна, группы и виджеты, а также настройки табличного

отображения.

2. Выполните аналогичные настройки для интерфейсов сущностей потомков.

Таким образом, у каждой сущности будет собственный набор настроек для отображения ее объектов на всех уровнях.

Структура `linkages`

Поле `linkages` представляет собой массив записей с описанием связей между сущностями.

В схеме домена поддерживаются линейные и древовидные связи.

Линейные связи

Линейные связи определяют отношения между сущностями в домене и позволяют строить направленные и ненаправленные связи различных типов.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Идентификатор связи.	Data ID	
`name`	нет	Пустая строка	Название связи.	строка	Поле с названием связи удобно использовать в случае, если сторону связи сложно определить из имеющихся данных, а поле `label` для элемента управления в представлении не определено.
`description`	нет	Пустая строка	Описание связи.	строка	Не выводится в представлениях. Отображается в схеме домена.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`side1`	да		Ссылка на первую связываемую сущность в паре связи.	Data ID Ref или FQID	Левая сторона действия в связи (направление связи идет от сущности, ссылка на которую указывается в этом поле). Хотя бы одна сущность в связи должна присутствовать в домене, в котором описывается связь.
`side2`	да		Ссылка на вторую связываемую сущность в паре связи.	Data ID Ref или FQID	Правая сторона действия в связи (направление связи идет к сущности, ссылка на которую указывается в этом поле). Хотя бы одна сущность в связи должна присутствовать в домене, в котором описывается связь.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`type`	да		Тип связи.	перечисление	Определяет тип отношений с каждой стороны связи. Доступные значения: • `1_1` — позволяет создавать пары связей двух сущностей, в которых ID объектов не повторяются. • `1_n` — позволяет создавать пары связей двух сущностей, в которых на одной стороне ID объектов могут повторяться, а на другой — нет. • `n_n` — позволяет создавать пары связей двух сущностей, в которых ID объектов могут повторяться на обеих сторонах.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`useForSearch`	нет	false	Индикатор использования данных связи в поисковых запросах.	перечисление	• `true` — данные этой связи и связанных объектов могут быть использованы в поисковых запросах. • `false` — данные этой связи и связанных объектов не могут быть использованы в поисковых запросах.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`undirected`	нет	`false`	Индикатор ненаправленной связи.	булево	Используется только при описании связи типа `n_n`. Доступные значения: • `true` — ненаправленная связь. Обе стороны одновременно осуществляют действие и подвергаются воздействию по отношению друг к другу. • `false` — направленная связь. Одна сторона является стороной действия, а другая — стороной воздействия.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
nameFrom	да		Глаголы связей для каждой из сторон.	объект	Определяет глагольное описание связи с каждой стороны, отображаемое в представлениях. Для любой направленной связи глагольные формы должны быть разными. См. описание поля nameFrom.
attributes	нет	Пустой массив	Пользовательские поля связи.	массив	Атрибуты, описывающие связанные сущности.

Структура nameFrom (в описании линейных связей)

Поле содержит описание глаголов связи.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
side1	да		Глагол связи с левой стороны.	строка	Глагольная форма действия. Например, "блокирует".
side2	да		Глагол связи с правой стороны.	строка	Глагольная форма воздействия. Например, "блокируется".

Древовидные связи

Древовидная связь позволяет организовать взаимодействие между объектами, при котором соединения образуют иерархическую структуру, напоминающую дерево. В этой структуре существует один корневой узел (главный элемент), от которого ответвляются дочерние узлы. Эти

узлы также могут иметь свои подчинённые элементы.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Идентификатор связи.	Data ID	
`name`	нет	Пустая строка	Название связи.	строка	
`description`	нет	Пустая строка	Описание связи.	строка	Не выводится в представлениях. Отображается в схеме домена.
`side1`	да		Ссылка на сущность, со стороны которой идет связь, в паре связи.	Data ID Ref или FQID	
`side2`	да		Ссылка на сущность, по направлению к которой идет связь, в паре связи.	Data ID Ref или FQID	
`type`	да		Тип связи.	перечисление	Для древовидной связи всегда указывается тип `1_n`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`nameFrom`	да		Глаголы связей для каждой из сторон.	объект	Определяет глагольное описание связи с каждой стороны, отображаемое в представлениях. Глагольные формы должны быть разными. См. описание поля `nameFrom`.

Структура `trees`

Поле `trees` содержит описание древовидной связи между сущностями.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Идентификатор дерева (уникален в рамках домена).	Data ID	
`type`	нет		Параметр, разрешающий изменять расположения узлов в этом дереве.	перечисление	Если изменение расположения узлов этого дерева разрешено, в поле устанавливается значение `orderable`. См. раздел Изменение расположения узлов в дереве.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
maxDepth	нет	Пустая строка	Максимально допустимое количество связей дерева от корневого узла до листового узла.	число	Определяет максимальную вложенность дерева.
hierarchy	нет	Пустая строка	Описание иерархии сущностей в дереве.	объект	См. описание поля hierarchy.

Структура hierarchy

Поле hierarchy содержит описание иерархии сущностей в дереве связей.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
entity	да		Ссылка на сущность в корневом узле.	Data ID Ref или FQID	Допустима ссылка на сущность в другом домене в формате FQDN/ID.
maxInstances	нет		Максимально допустимое количество объектов сущности в корневом узле.	число	Ограничивает создание объектов одной сущности в дереве.
maxChildren	нет		Максимально допустимое количество дочерних узлов для одного родительского узла через указанную связь.	число	Ограничивает создание связей между сущностями в корневом узле и дочерних узлах.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`children`	нет		Описание объектов, дочерних к сущности, на которую ссылается поле entity.	массив	См. описание поля `children`.

Структура `children`

Поле `children` содержит описание объектов, дочерних к сущности, на которую ссылается поле `entity` в блоке `hierarchy`.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`linkage`	да		Ссылка на идентификатор связи, описанной в поле `linkages`	Data ID Ref или FQID	Необходимо сослаться на связь, описывающую отношения между сущностями в корневом узле и дочерних узлах.
`entity`	да		Ссылка на идентификатор сущности, дочерней к сущности в корневом узле.	Data ID Ref или FQID	
`maxInstances`	нет		Максимально допустимое количество объектов одной сущности в разделе.	число	Это значение не может превышать значение параметра `maxChildren` в описании сущности в корневом узле.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`maxChildren`	нет		Максимально допустимое количество дочерних узлов для сущности, указанной в поле `entity`, через связь, указанную в поле `linkages`, в этом блоке.	число	Ограничивает количество связей между сущностями в поле `entity` и в дочерних узлах.
`side`	нет		Указывает на сторону связи, с которой находится родительская сущность.	число	По умолчанию принимает значение 1.

Вы можете создать в схеме домена несколько вложенных блоков `children` для описания узлов ветвления разных уровней. Предварительно убедитесь, что значение параметра `maxDepth` в блоке `trees` позволяет это сделать.

Структура `dataTypes`

В схеме домена можно определять пользовательские типы данных для атрибутов типов данных, доступных в системе. С их помощью можно описывать перечисления, рабочие процессы и таблицы.

Поля `dataTypes` включают в себя:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Идентификатор типа данных.	Data ID	
`name`	нет	Пустая строка	Название типа данных.	строка	
`description`	нет	Пустая строка	Описание элемента домена.	строка	Не выводится в представлениях. Отображается в схеме домена
`dataType`	да		Описываемый тип данных.	перечисление	См. статью Типы данных в схемах доменов.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`entity`	да (для `Reference`)		Ссылка на сущность.	Data ID Ref или FQID	Поле доступно только для типа данных `Reference`.
`constraints`	нет	Пустой массив	Ограничители типа данных атрибута.	массив ограничителей типа данных	Для каждого типа данных свой набор ограничителей со своим набором полей.
`values`	да (для перечислений)		Массив записей, входящих в перечисление.	массив	Доступно для перечислений, см. описание поля `values`.
`view`	нет	Пустой массив	Представление значений перечисления, статусов рабочего процесса, или табличное представление.	массив	Доступно для перечислений, рабочих процессов и таблиц, см. описание поля `view`.
`statuses`	да (для рабочих процессов)		Статусы процесса.	массив	Доступно для рабочих процессов. Новые статусы в существующий рабочий процесс можно добавлять в любое место списка. Удалять или менять местами существующие статусы нельзя. См. описание поля `statuses`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`initial`	да (для рабочих процессов)		Начальный статус процесса.	строка	Доступно для рабочих процессов.
`transitions`	да (для рабочих процессов)		Допустимые переходы между статусами.	массив	Доступно для рабочих процессов, см. описание поля `transitions`.
`attributes`	да (для таблиц)		Массив атрибутов, которые могут отображаться в столбцах таблицы.	массив	Доступно для таблиц, см. описание поля `attributes`.
`maxRows`	нет		Максимальное количество записываемых строк в таблице.		Доступно для таблиц.
`sorting`	нет		Правило сортировки.		См. описание поля `sorting`.

Структура `values`

Поле `values` включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Идентификатор значения перечисления.	строка	
`name`	нет	Пустая строка	Наименование значения перечисления.	строка	
`description`	нет	Пустая строка	Описание элемента домена.	строка	Не выводится в представлениях. Отображается в схеме домена

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`hidden`	нет	false	Скрытие значения.	булево	Если установлено значение `true`, указанное значение становится недоступным (скрывается) для записи в атрибуты объектов в любых операциях для любых пользователей. Для всех операций чтения и применения в фильтрах скрытое значение по-прежнему остается доступным.

Структура `view`

Поле `view` используется для описания представления значений, а также цветов, с помощью которых отображаются статусы рабочего процесса.

Поле	Обязательное поле	Описание	Тип данных	Комментарий
`value`	нет	Ссылка на значение перечисления.	Data ID Ref	Доступно для перечислений.
`status`	да (для рабочих процессов)	Ссылка на значение перечисления.	Data ID Ref	Доступно для рабочих процессов.

Поле	Обязательное поле	Описание	Тип данных	Комментарий
`color`	нет	Цвет значения.	строка	В этом поле можно задать цвета для требуемых значений перечисления или статусов (там, где это необходимо). Цвета задаются согласно схеме CSS. Можно использовать следующие обозначения цвета: • название (например, `blue`) • HEX-код (например, `#0000ff`). Если задать код в шаблоне #rgb (например: `#00f`), система автоматически распознает его как `#rrggbb`.
`columns`	да (для таблиц)	Массив описаний столбцов таблицы.	массив	Доступно для таблиц, см. описание поля `columns`.
`sortingBy`	нет	Описание правила сортировки.	массив	Доступно для таблиц, см. описание поля `sortingBy`.
`paging`	нет	Настройки количества записей, показываемых в таблице.		Доступно для таблиц, см. описание поля `paging`.

Структура `columns` (в таблице представлений значений)

Поле `columns` используется для описания столбцов таблицы.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Идентификатор столбца таблицы.	массив	
`attribute`	да		Атрибут столбца.	строка	
`label`	нет	Пустая строка	Лейбл столбца.	строка	В этом поле можно задать произвольное наименование столбца таблицы. Если поле не задано, система использует в качестве наименования значение поля `id` столбца таблицы.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`control`	нет		Тип редактора, используемый в столбце, в соответствии с таблицей.	строка	Определяется системой автоматически на основе указанного атрибута. В этом поле можно изменить используемый по умолчанию тип редактора в столбце на другой допустимый тип или задать параметры редактора. Допустимые типы для разных контекстов представлены в таблице.
`layout`	нет		Массив настроек размера и видимости столбца.	массив	См. описание поля `layout`.

Структура `layout` (в описании столбцов таблицы)

Поле `layout` содержит массив настроек размера столбца.

Настройка включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`width`	нет		Ширина столбца (в пикселях).	объект	См. описание поля `width`.

Настройки ширины столбца (`width`)

Поле `width` содержит массив настроек ширины столбца.

Настройка включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных
`default`	да		Ширина столбца по умолчанию, в пикселях.	целое положительное число
`min`	да		Максимальная ширина столбца в пикселях.	целое положительное число
`max`	да		Минимальная ширина столбца в пикселях.	целое положительное число

Структура `statuses`

Поле `statuses` используется для создания массива записей статусов рабочего процесса.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`			Идентификатор статуса.	Data ID	
`name`	нет	Пустая строка	Название статуса.	строка	
`description`	нет	Пустая строка	Описание элемента домена.	строка	Не выводится в представлениях. Отображается в схеме домена.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`hidden`	нет	false	Скрытие значения.	булево	Если установлено значение `true`, указанное значение становится недоступным (скрывается) для записи в атрибуты объектов в любых операциях для любых пользователей. Для всех операций чтения и применения в фильтрах скрытое значение по-прежнему остается доступным.

Структура `transitions`

Поле `transitions` используется для создания массива записей допустимых переходов между статусами рабочего процесса.

Поле	Описание	Тип данных	Комментарии
`from`	Идентификатор статуса, для которого описывается переход.	строка	Значение * позволяет задать переход из любого статуса.
`to`	Идентификатор статуса, в который может перейти исходный статус.	строка	

Нотация и стереотипные поля в схеме домена

Таблица нотации

Идентификатор	Правила
Домен	Соответствуют стандартным соглашениям об именовании для доменных имен DNS. Пример: `common.domain.tld` Ознакомьтесь с описанием FQID.
Сущности (Data ID)	Все слова пишутся с заглавной буквы. Разделителей между составными частями нет. Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`). Длина не более 30 символов. Примеры: • `Asset` • `DeviceConfiguration`
Атрибуты (Data ID)	Первое слово пишется со строчной буквы, следующие — с заглавной буквы. Разделителей между составными частями нет (кроме атрибута типа `Reference`). Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`). Длина не более 30 символов. Примеры: • `os`; • `updatedAt`
Типы данных (Data ID)	Все слова пишутся с заглавной буквы. Разделителей между составными частями нет. Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`). Длина не более 63 символов. Примеры: • `String`; • `Uuid`; • `MyTable`.

Идентификатор	Правила
Представления, группы, действия, виджеты (View ID)	Первое слово пишется со строчной буквы, следующие — с заглавной буквы. Разделителей между составными частями нет. Можно добавлять произвольный префикс для обозначения группы (например, `buttons`, `btns`, `actions`, `act`, `widgets`, `forms`). Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`) и точка (`.`). Для других идентификаторов точку использовать нельзя, так как в этом случае точка будет являться специальным символом. Длина не более 63 символов. Примеры: • `deviceView` (вариант: `lists.device`); • `deviceCreateButton` (вариант: `buttons.createDevice`); • `deviceCreateAction` (вариант: `actions.createDevice`).
Перечисления константных значений (`type`, `kind`)	Слова разделяются символами подчеркивания. Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`). Длина не более 63 символов. Примеры: • `abstract`; • `open_modal`.
Пути в меню и группах меню	Стандартный путь к местоположению ресурса URL. Примеры: • `/assets`; • `/assets/main-devices`.
Медиаресурсы (иконки, изображения)	Первое слово пишется со строчной буквы, следующие — с заглавной буквы. Разделителей между составными частями нет. Можно добавлять префикс для обозначения группы (`icons`, `images`). Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`). Длина не более 63 символов. Примеры: • `iconsassets`; • `iconsdevice`.

Ограничения нотации являются формальными и применяются внутри исходных схем доменов, поставляемых системой. Они могут не соблюдаться, если пользователь создает свою собственную схему домена. Все идентификаторы

являются регистрозависимыми.

Определение доменов, сущностей и атрибутов в схеме домена

Data ID — идентификатор элемента хранения. Не может быть равен имени любого системного поля или зарезервированному имени устройства Windows.

View ID — идентификатор элемента представления. Не может быть равен зарезервированному имени устройства.

▼ Системные поля объектов и связей в доменах

- `id` — идентификатор объекта или связи в текущем домене.
- `organisationId` — идентификатор организации.
- `domainId` — идентификатор домена из схемы домена.
- `entityId` — идентификатор сущности из схемы домена.
- `createdAt` — дата и время создания объекта или связи.
- `createdBy` — пользователь, который создал объект или связь.
- `updatedAt` — дата и время последнего обновления объекта или связи.
- `updatedBy` — пользователь, который в последний раз обновил объект или связь.
- `tenantId` — тенант, который является владельцем объекта.
- `label` — человекопонятный идентификатор объекта для пользователя.
- `version` — версия объекта.

▼ Зарезервированные имена устройств

- aux
- com1
- com2
- com3
- com4
- com5
- com6
- com7
- com8
- com9
- lpt1
- lpt2
- lpt3
- lpt4
- lpt5
- lpt6

- lpt7
- lpt8
- lpt9
- con
- nul
- prn

Data ID Ref — ссылка на Data ID элемента хранения домена, который уже определен в другом месте. По ссылке система может получить все необходимые метаданные.

View ID Ref — ссылка на View ID элемента хранения домена, который уже определен в другом месте. По ссылке система может получить все необходимые метаданные.

Data ID Ref и View ID Ref — короткие формы полного идентификатора. Они всегда являются локальным идентификатором по отношению к текущей схеме домена. Для большинства элементов схемы домена вместо короткой формы можно указывать полный идентификатор — FQID, который может ссылаться на другой домен.

Домены, сущности и атрибуты определяются в схеме домена при помощи **FQID** (Fully Qualified ID — полностью определенного идентификатора) согласно следующим правилам:

- **Домен**: FQID домена определяется как его FQDN (Fully Qualified Domain Name — полностью определенное доменное имя).
- **Сущность**: FQID сущности определяется как FQDN/ID сущности. Если сущность определена внутри текущего домена, можно опустить FQDN и указать просто ID сущности.
- **Атрибут**: FQID атрибута какой-либо сущности определяется как **FQID сущности/ID атрибута**:
  - Если сущность определена внутри текущего домена, можно опустить FQID сущности и указать просто **ID атрибута**.
  - Если сущность определена в другом домене, необходимо сослаться на домен: **FQDN/ID сущности.ID атрибута**.
  - Если атрибут относится к типу Reference, его FQID определяется как: **FQID атрибута-ссылки.ID атрибута из проекции**.

Редакторы и представления атрибутов

В таблице ниже указано, какие элементы управления могут использоваться в настройках виджета в поле `control` для типов данных атрибутов в том или ином контексте.

Если в настройках виджета указан элемент, отсутствующий в поле таблицы для этого контекста и типа данных, по умолчанию используется элемент из этого поля. Если в поле указано несколько элементов, по умолчанию используется первый элемент.

Тип данных атрибута	Редактор в форме значения атрибута (модальное окно)	Редактор значения атрибута по месту (панель)	Нередактируемое предзаполненное представление значения атрибута (модальное окно)	Представление значения атрибута (панель)	Представление значения атрибута (ячейка таблицы)
`Uuid`	uuid		uuid	text	text
`Timestamp`	timestamp	timestamp	timestamp	text	text
`Date`	date	date	date	text	text
`Time`	time	time	time	text	text
`TimestampRange`	timestamp_range		timestamp_range	text	text
`String`	text, text_area	text, text_area	text, text_area	text, text_area	text
`MAC`	mac	mac	mac	text	text
`Integer`	integer, integer_slider	integer	integer, integer_slider	text	text
`Decimal`	decimal, decimal_slider	decimal	decimal, decimal_slider	text	text
`Float`	float, float_slider	float	float, float_slider	text	text
`Bool`	switch, checkbox		switch, checkbox	switch, checkbox, bool_text	switch, checkbox, bool_text
`Enum`	select, radio	select	select, radio	tag, level	tag, level
`IpAddress`	ip	ip	ip	text	text
`CIDR`	cidr	cidr	cidr	text	text
`Attachment`	file		file	file, text	file, text
`Reference`	link, tag		link, tag	link, tag	link, tag
`Workflow`	select	select	select	select, level	tag, level

Таблица 13. Описание некоторых редакторов

Название	Комментарии
`checkbox`	флажок
`tag`	тег
`ip`	поле для ввода IP-адреса
`level`	поле для выбора уровня
`link`	ссылка

Название	Комментарии
`radio`	радиокнопка
`slider`	ползунок Типы ползунка: • `float_slider` — в качестве чисел шкалы используются числа с плавающей точкой; • `integer_slider` — используются целые числа; • `decimal_slider` — используются числа с заданной точностью.
`select`	выпадающий список
`switch`	переключатель
`text_area`	поле для ввода текста

Системные атрибуты

В ходе своей работы система для каждого объекта по умолчанию создает набор системных атрибутов и автоматически заполняет их значения.

Обычные (несистемные) атрибуты задаются для объекта — в поле `attributes` схемы домена. В отличие от них системные атрибуты **не нужно** специально задавать в схеме домена. Они уже созданы системой.

На системные атрибуты **объектов** можно ссылаться при описании виджетов атрибутов. Ссылка на системный атрибут в виджете атрибутов выглядит точно так же, как ссылка на обычный атрибут. Вы можете отображать системные атрибуты в интерфейсах объектов и списков объектов — там, где это нужно пользователю.

У каждого системного атрибута есть неизменяемое название на русском языке (см. в таблице). Это название системного атрибута будет отображаться в виджете атрибута в пользовательском интерфейсе.

При использовании системного атрибута в списочном представлении необходимо помнить, что можно отредактировать название столбца, где отображается такой атрибут, а название системного атрибута в виджете атрибута отредактировать нельзя.

В таблице ниже приведен список существующих системных атрибутов объекта, на которые вы можете ссылаться при создании виджетов атрибутов:

Название системного атрибута объекта	Тип данных	Описание	Отображаемое название атрибута
`id`	UUID	Идентификатор объекта в текущем экземпляре домена	Идентификатор объекта

Название системного атрибута объекта	Тип данных	Описание	Отображаемое название атрибута
`organizationId`	Reference (Tenant)	Отображаемое название организации, присвоенное объекту. В схеме домена определяется необходимость отображения виджета атрибута в модальном окне. • Если виджет отображается, пользователь может выбрать доступную организацию. • Если пользователь не выбирает организацию, система автоматически считает объект справочником.	Организация
`domainId`	String (FQDN)	Идентификатор домена из схемы домена	Идентификатор домена
`entityId`	String (Definition ID)	Идентификатор сущности из схемы домена	Идентификатор сущности
`createdAt`	Timestamp	Дата и время создания объекта с точностью до секунд	Дата создания
`createdBy`	Reference (User)	Отображаемое имя пользователя, который создал объект.	Создал
`updatedAt`	Timestamp	Последняя дата и время обновления объекта с точностью до секунд	Дата изменения
`updatedBy`	Reference (User)	Отображаемое имя пользователя, который последний раз обновил объект.	Изменил

Название системного атрибута объекта	Тип данных	Описание	Отображаемое название атрибута
`tenantId`	Reference (Tenant)	Отображаемое название тенанта, который является владельцем объекта.	Идентификатор тенанта
`label`	String	Неуникальный человекочитаемый идентификатор объекта, предназначенный для пользователя	Лейбл
`version`	Integer	Версия объекта	Версия объекта

Составление схемы домена

Для создания схемы домена:

1. Укажите в метаданных схемы информацию, необходимую для создания домена (функциональной области).
2. Опишите сущности, которые будут отображаться в домене на основе схемы.

   Например, домен **Активы** с сущностями **Оборудование** и **Пользователи**.

3. Для сущностей опишите атрибуты, которыми характеризуется сущность.

   Например, у сущности **Оборудование** могут быть атрибуты **Имя**, **IP-адрес**, **Домен**.

4. При необходимости задайте пользовательские типы данных для атрибутов с использованием типов данных, доступных в системе. С их помощью можно описать перечисления, таблицы и рабочие процессы.
5. Опишите параметры сортировки данных при работе с описанием интерфейса списка объектов.
6. Если необходимо автоматически подбирать дополнительные данные для хранимых ссылок на определенные объекты, опишите проекции для этих сущностей.
7. Задайте связи для сущностей, которые будут отображаться в домене на основе схемы.
8. Если сущность является потомком другой сущности, задайте параметры наследования.
9. Опишите интерфейс объекта:
   a. Опишите действия, которые будут выполняться над объектами.
   b. Опишите иконки для виджетов представлений и элементов меню, используя медиаресурсы из библиотеки.
10. Опишите структуру меню для раздела домена и его сущностей, отображаемого в главном меню системы — название пунктов меню и параметры их отображения, группировку элементов меню и ссылки на используемые иконки.
11. Опишите виджеты, которые будут отображаться в представлениях.

12. Опишите элементы меню, отображаемые во вкладках и панелях.
13. Опишите панели и модальные окна, которые будут отображаться для экземпляров сущности.

    Опишите визуальные элементы, которые будут включаться в панели и модальные окна. К ним относятся вкладки, состоящие из блоков, а также формы создания и редактирования сущностей.

14. Опишите интерфейс списка объектов:
    a. Укажите ссылки на идентификаторы действий, которые будут выполняться со списком объектов (например, массовые действия удаления, импорта и экспорта).
    b. Укажите ссылки на виджеты действий, которые будут отображаться в панели действий интерфейса (например, кнопки удаления, импорта и экспорта).
    c. Сформируйте список атрибутов сущности, которые будут отображаться в табличном отображении.
    d. Укажите настройки ширины и видимости столбцов таблицы.
    e. Укажите параметры табличного отображения: фильтрация, сортировка и группировка по столбцам, количество показываемых записей, настройки обновления данных, настройки возможности выбора строк.

Конфигурация рабочего процесса

В схеме домена можно задать рабочий процесс, доступные статусы процесса и возможность перехода между статусами. В настройках процесса должны быть указаны как минимум два статуса: начальный и конечный.

В компактном представлении домена в поле **Статус** отображается текущий статус рабочего процесса.

При попытке перехода в другой статус в ходе рабочего процесса система может отображать следующую информацию:

- Уведомление об ошибке — если для перехода необходимо загрузить файл или ввести данные.
- Информационное уведомление — если для перехода необходимо получить обновленные данные сущности. Отображается окно с кнопкой обновления данных.

В поле **Статус** в компактном представлении домена отображается текущий статус рабочего процесса.

Пользовательские типы данных

В схеме домена можно определять пользовательские типы данных для атрибутов с использованием типов данных, доступных в системе. С их помощью можно описывать перечисления, рабочие процессы и таблицы.

Компактное представление сущности

Настройки компактного представления сущности определяют отображение детальной информации о выбранном экземпляре сущности.

Действия

При добавлении схемы домена для каждой его сущности можно выбрать действия с экземплярами сущности, которые будут доступны в разделе сущности. Названия кнопок, инициирующих действия, задаются в схеме произвольно.

В системе доступны следующие действия:

- добавление экземпляра сущности;
- редактирование атрибута экземпляра сущности непосредственно в окне системы (без вызова отдельного окна редактирования);
- удаление экземпляра сущности;
- редактирование атрибутов экземпляра сущности в отдельном окне редактирования;
- запуск плейбука из списка;
- открытие в соседней вкладке списочного представления с контекстом.

Множественный выбор экземпляров сущностей

В списочном представлении доступен множественный выбор экземпляров сущностей. К нему можно привязать действие массового удаления экземпляров сущностей, которое определяется в схеме домена.

Сортировка экземпляров сущностей

В списочном представлении доступна сортировка экземпляров сущностей по столбцам таблицы, определенным в правилах сортировки. В таблице можно отображать не все атрибуты, которые участвуют в сложном правиле сортировки — система все равно будет отображать экземпляры, отсортированные по таким атрибутам. Для удобочитаемости таблицы рекомендуется отображать первый атрибут, участвующий в правиле сортировки.

Правила сортировки экземпляров сущностей определяются в схеме домена. Если хотя бы одно правило сортировки не определено в схеме домена, система сортирует экземпляры в таблице по времени поступившей о них информации за секундный интервал и далее сортирует их по идентификатору экземпляра.

Группировка экземпляров сущностей

В списочном представлении доступна группировка экземпляров сущностей по столбцу таблицы, определенному в правиле группировки.

Сгруппированные данные в таблице по умолчанию отсортированы согласно правилу сортировки. Правила группировки экземпляров сущностей определяются в схеме домена.

Работа с организациями в списочном представлении

В схеме домена можно отметить системный атрибут `organizationid` как обязательный для заполнения и использовать его в представлениях объектов. Если виджет этого атрибута используется в модальном окне создания объектов, то он примет вид раскрывающегося списка с лейблом виджета. Значения списка зависят от тенанта, в рамках которого работает пользователь.

Значения атрибута `organizationid` задаются только при создании объекта.

Если для работы в системе выбран **Основной тенант**, в раскрывающемся списке отображаются организации, соответствующие дочерним тенантам.

Если для работы в системе выбран какой-либо дочерний тенант, в раскрывающемся списке отображается организация, соответствующая этому тенанту.

Если атрибут `organizationid` не заполняется пользователем, объект будет доступен для отображения во всех списках.

Фильтрация и определение контекстов списочного представления в рамках тенанта

Система всегда выбирает для отображения объекты, исходя из того, в каком тенанте сейчас работает пользователь.

Последующие фильтры и контексты применяются к итоговой выборке объектов.

Описания интерфейса

Интерфейс для работы с объектами позволяет просматривать и редактировать информацию об объектах, работать с различными визуальными элементами, а также выполнять действия с объектами. Описание интерфейса системы является частью схемы домена и включается в поле `views` на уровне домена.

В описаниях интерфейса используются ссылки на действия, меню домена, меню панелей, а также иконки для виджетов представлений и элементов меню.

Структура `actions`

Поле `actions` представляет собой массив записей действий. Это необязательное поле.

Запись действия включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Уникальный идентификатор действия.	View ID	
`name`	нет	Пустая строка	Наименование действия.	строка	Наименование действия используется, если не определен или не может быть определен `label` для элемента управления в представлении.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`description`	нет	Пустая строка	Описание действия.	строка	Не выводится в представлениях. Отображается в схеме домена.
`type`	да		Тип действия.	перечисление	См. список доступных действий.
`modal`	да		Ссылка на модальное окно, которое будет открываться при этом действии.	строка	Доступно только для действия с типом `open_modal`
`panel`	да		Ссылка на панель, которая будет открываться при этом действии.	строка	Доступно только для действия с типом `open_panel`
`view`	да		Ссылка на списочное представление объектов, зависимых от контекста.	View ID Ref	Доступно только для действия с типом `open_list`
`entity`	да		Ссылка на сущность, над экземпляром которой производится действие.	Data ID Ref или FQID	Доступно только для действий с типами: • `remove`; • `workflow`; • `open_panel`; • `open_modal`; • `open_graph_view`; • `bulk`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`operation`	да		Тип операции над объектом.		Для типа действия `open_modal` доступны следующие типы операций: • `create` — добавление объекта; • `edit` — изменение выбранного объекта. Для типа действия `bulk` доступны следующие типы операций: • `remove` — удаление выбранных объектов; • `edit_attribute` — изменение атрибута в выбранных объектах с типом данных `workflow`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`playbook`	нет		Ссылка на плейбук или скрипт.	строка	Уникальное наименование плейбука или скрипта в рамках текущей экспертизы. Доступно только для действия с типом `playbook`.
`contextPin`	нет		Запрет удаления контекста из фильтра списка объектов.	булево	Если есть запрет на удаление контекста, в ярлыке контекста в списочном представлении отсутствует крестик, позволяющий удалить контекст. Доступные значения: • `true`; • `false`. Доступно только для действия с типом `open_list`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`page`	нет		Окно, в котором будет отображен список.	перечисление	Открываемый список может быть отображен как в новом окне, так и в текущем. Возможные значения: • `blank` — показать список в новом окне; • `current` — показать список в текущем окне. Доступно только для действия с типом `open_list`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`attribute`	нет		Ссылка на атрибут-ссылку, для которого будет применен контекст.		Список объектов можно отфильтровать по значению атрибута-ссылки сущности этих объектов. Значение устанавливается из контекста при открытии списка. Указывать одновременно атрибут-ссылку и связь в описании действия нельзя. Доступно только для действий с типом `open_list` и `bulk`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`linkage`	нет		Ссылка на связь, для которой будет применен контекст.		Список объектов можно отфильтровать по одной из сторон связи этих объектов. Значение устанавливается из контекста при открытии списка. Указывать одновременно атрибут-ссылку и связь в описании действия нельзя. Доступно только для действия с типом `open_list`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`side`	нет		Сторона ссылки или связи, в соответствии с которой отображаются объекты в списочном представлении с контекстом.	Data ID Ref или FQID	Доступные значения: • `1` — первая сторона связи; • `2` — вторая сторона связи. В списочном представлении отобразятся объекты указанной стороны связи. Указывается только для симметричной направленной связи.

Доступные типы действий

Доступны следующие типы действий:

- `open_panel` — открыть панель деталей объекта (только для панели);
- `open_modal` — открыть модальное окно для создания или редактирования объекта (только для модального окна);
- `open_list` — открыть списочное представление объектов с заполненным контекстом на основании выбранного объекта в текущем списке;
- `open_graph_view` — открыть граф связей, позволяющий создавать, изменять и удалять экземпляры связей, а также просматривать связи объекта на разных уровнях;
- `remove` — удалить объект;
- `bulk` — массовое действие;
- `playbook` — запуск выполнения указанного плейбука или скрипта;
- `generate_qr_code` — открыть модальное окно со сгенерированным QR-кодом, содержащим ссылку на компактное представление сущности.

Структура `menus`

Поле `menus` представляет собой массив записей элементов меню.

Запись включает следующие поля:

Поле	Обязательное поле	Описание	Тип данных	Комментарий
`id`	да	Идентификатор элемента меню.	View ID	
`route`	да	Компонент адреса текущего пункта меню, отображаемый в строке веб-браузера.	URL path	Фактически определяет путь в адресной строке, по которому можно перейти на целевую страницу. Пути должны быть уникальными как в рамках одной схемы домена, так и в разных схемах.
`label`	да	Отображаемое в меню системы название раздела домена.	строка	
`icon`	нет	Ссылка на медиа-ресурс.	строка	View ID Ref

Поле	Обязательное поле	Описание	Тип данных	Комментарий
`type`	да	Тип элемента меню.	перечисление	Тип `root_group` используется для отображения в меню системы корневой группы элементов меню, включающей названия раздела домена и подразделов сущностей. Вы можете создавать любое количество корневых групп элементов меню. Тип `root_menu` используется для отображения в меню системы элемента меню, не включенного в корневую группу элементов меню.
`view`	да	Ссылка на представление.	View ID Ref	Ссылка на списочное представление или на представление списков с вкладками. Поле доступно только для типов элементов меню: • `root_menu`; • `menu`
`description`	нет	Описание меню.	строка	Не выводится в представлениях. Отображается в схеме домена

Поле	Обязательное поле	Описание	Тип данных	Комментарий
`placement`	да	Настройки расположения меню в левой панели.	перечисление	Определяет, в каком меню разместить указанный элемент или группу. Доступны следующие значения: • `top` — меню будет отображаться в главном меню системы; • `settings` — меню будет отображаться в разделе **Настройки** главного меню системы.
`items`	да	Дочерние элементы для корневой группы меню.	массив	Массив записей меню с подразделами типов сущностей. Используется для типа элементов меню `root_menu`. См. описание поля `items`.

Структура `items`

Поле `items` используется для создания массива записей пунктов меню в разделе домена.

Запись включает следующие поля:

Поле	Обязательное поле	Описание	Тип данных	Комментарий
`id`	да	Идентификатор элемента меню.	View ID	

Поле	Обязательное поле	Описание	Тип данных	Комментарий
`route`	да	Компонент адреса текущего пункта меню, отображаемый в строке веб-браузера.	URL path	Фактически определяет путь в адресной строке, по которому можно перейти на целевую страницу.
`label`	да	Отображаемое в меню системы название пункта меню.	строка	
`type`	да	Тип элемента меню.	перечисление	Устанавливается значение `menu` — дочерний элемент меню, который может быть задан только в рамках массива подэлементов корневой группы.
`view`	да	Ссылка на представление.	View ID Ref	Ссылка на списочное представление или на представление списков с вкладками.
`description`	нет	Описание меню.	строка	Не выводится в представлениях. Отображается в схеме домена.

Структура `medias`

Поле `medias` представляет собой массив записей элементов иконок.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Идентификатор иконки, которая будет использоваться в виджетах представлений и элементах меню.	View ID	
`description`	нет	Пустая строка	Описание элемента домена.	строка	Не выводится в представлениях. Отображается в схеме домена
`type`	да		Тип иконки.	перечисление	Определяет тип медиа ресурса и его возможности. В текущей реализации поддерживается один тип иконки `icon_name` — именованная иконка из дизайн-системы.
`iconName`	да		Идентификатор медиаресурса в библиотеке	строка	

Структура `views`

Описание интерфейса указывается в поле `views` и содержит следующие разделы:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`id`	да		Идентификатор интерфейса объекта.	View ID	

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`type`	да		Описание интерфейса объекта.	перечисление	Определяет тип интерфейса, который влияет на его поведение и возможности с учетом контекста использования. См. список доступных типов.
`label`	нет	Пустая строка	Наименование.	строка	Наименование представления для заголовка представления и при переиспользовании в смешанном представлении.
`description`	нет	Пустая строка	Описание элемента домена.	строка	Не выводится в представлениях, а отображается на визуализации схемы домена.
`entity`	да (для объектов)		Ссылка на идентификатор объекта.	строка	Доступно для интерфейса объектов.
`widgets`	да (для объектов)		Массив записей виджетов.	массив	Доступно для интерфейсов объектов и дашбордов.
`groups`	да (для объектов)		Массив записей групп.	массив	Доступно для интерфейса объектов.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии
`views`	да (для объектов)		Массив записей панелей и модальных окон.	массив	Доступно для интерфейса объектов.
`source`	да (для списка объектов)		Источник списка объектов.	источник списка объектов	Доступно для интерфейса списка объектов. См. описание поля `source`.
`actionPanel`	нет	Пустой массив	Содержит ссылки на виджеты, используемые в описании (описываются в настройках интерфейса объекта).	массив ссылок на виджеты	Доступно для интерфейса списка объектов. Ссылки могут указывать на виджеты действий, которые могут работать в любом контексте. Если действие может работать только в контексте одного объекта, такие виджеты будут недоступны, пока не будет выбран объект в списке.
`table`	да (для списка объектов)		Настройки табличного отображения в интерфейсе списка объектов.	таблица списка объектов	Доступно для интерфейса списка объектов. См. описание поля `table`.
`defaultTab`	нет	Первая указанная вкладка	Вкладка, отображаемая открытой по умолчанию.	View ID Ref	Доступно для группы вкладок со списками объектов.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарии

`tabs`	да (для группы вкладок)		Массив ссылок на идентификаторы описаний интерфейсов списков.
			
			Формат поля:
			
			`tabs:`
			`  - view: <ссылка на описание интерфейса списка 1>`
			`  - view: <ссылка на описание интерфейса списка 2>`
				массив ссылок на вкладки	Доступно для группы вкладок со списками объектов. Каждая ссылка должна указывать на группу представлений только с типом `tab`.

`layout`	нет (для интерфейса виджетов дашбордов)		Настройки взаимного расположения компонентов в представлении.	размещение виджетов представления	См. описание поля `layout`.

Доступные типы интерфейсов

В схеме домена доступны следующие типы интерфейсов:

- `entity` — представления одного объекта указанной сущности для различных операций и контекста.
- `list` — представление списка объектов для различных контекстов.
- `tabs_list` — группа вкладок, в каждой из которых отображается независимое представление списка объектов.
- `dashboard` — группа плиток на странице, позволяющая запускать выполнение определенных действий.
- `tree` — представление древовидных связей.

Структура `source`

Поле `source` представляет собой параметры источника объектов для списка.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`type`	да	operational	Тип источника.	перечисление	Определяет метод запроса данных, поведение и возможности интерфейса. Всегда устанавливается значение `operational` (база данных системы).
`entity`	да		Ссылка на сущность, данные объектов и связей которой отображаются в списке.	Data ID Ref, FQID	
`joins`	нет		Описание связей с сущностями, данные которых отображаются в списке.	массив	См. описание поля `joins`.

Структура `joins` в параметрах источника объектов для списка

Поле `joins` представляет собой массив записей связей с сущностями, данные которых будут отображаться в списке.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Идентификатор связи.	Data ID	
`linkage`	да		Ссылка на связь с сущностью, данные которой будут отображаться в списке.	Data ID Ref	

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`side`	нет		Сторона связи.	перечисление	Доступные значения: • `side 1` • `side 2` Сторона определяется только для симметричной направленной связи.
`description`	нет	Пустая строка.	Описание связи.	строка	

Структура `widgets`

Поле `widgets` представляет собой массив записей виджетов.

Доступные типы виджетов

В схеме домена доступны следующие типы виджетов:

- `attribute` — виджет атрибута, позволяющий отображать и редактировать значение атрибута объекта.
- `action` — виджет действия, позволяющий запускать действие с передачей на сторону исполнения текущего контекста.
- `linkage` — виджет связи, позволяющий отображать и менять список связанных объектов.
- `visualization` — виджет графического отображения, позволяющий отображать графики.
- `history` — виджет истории изменений, позволяющий отображать события аудита, связанные с объектом.
- `comments` — виджет комментариев, позволяющий добавлять комментарии, связанные с объектом и отображать существующие комментарии.
- `linkage_list` — виджет списка связей, позволяющий отображать связи объекта для просмотра и редактирования согласно заданным правилам доступности связей в доменах.
- `card` — виджет дашбордов, позволяющий запускать в дашбордах действия над сущностями домена, описанные в поле `actions`.

Запись виджета включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Уникальный идентификатор виджета	View ID	
`type`	да		Тип виджета	перечисление	Тип виджета, который влияет на поведение интерфейса с учетом контекста использования. См. раздел Доступные типы виджетов.
`attribute`	да (для виджетов атрибутов)		Ссылка на атрибут	Data ID Ref	Доступно для виджетов атрибутов, атрибут сущности, значение которого будет отображаться и редактироваться для выбранного объекта.
`action`	да (для виджетов действий)		Ссылка на действие над объектом	View ID Ref	Доступно для виджетов действий, если не задано поле `actions`. Действие должно быть предварительно описано. В описании действия нельзя одновременно использовать поля `action` и `actions`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`actions`	да (для виджетов действий)		Список ссылок на действия	массив ссылок на действия	Доступно для виджетов действий, если не задано поле `action`. Список действий, которое будет формировать меню выбора действия при нажатии на элемент управления виджета. В описании действия нельзя одновременно использовать поля `action` и `actions`. См. описание поля `actions`.
`linkage`	да (для виджетов связей)		Ссылка на связь	Data ID Ref	Доступно для виджетов связей, связь сущности, состав которой для выбранного объекта будет отображаться и редактироваться.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`side`	нет	`2`	Сторона связи для виджета	перечисление	Доступно для виджетов связей, сторона определяется только для симметричной направленной связи. В других случаях сторону определять не нужно: ее значение будет игнорироваться. Возможные значения: • `1` — виджет отображает сторону 1, а объект компактного представления находится на стороне 2. • `2` — виджет отображает сторону 2, а объект компактного представления находится на стороне 1.
`label`	нет	Пустая строка	Наименование	строка	Доступно для виджетов связей, наименование связи, отображаемое в виджете.
`description`	нет		Описание виджета	строка	

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`control`	нет		Элемент управления	элемент управления	Определяется автоматически на основании заданного атрибута, действия или связи, на которые ссылается виджет. Для виджетов атрибутов и связей можно задать свой элемент управления. Элемент должен соответствовать типу атрибута или типу и направленности связи. Для виджетов действий можно задать иконку (кроме случаев, когда в описании действия указаны значения полей `type = bulk` и `operation = edit_attribute`). См. описание поля `control`.
`name`	нет		Название визуализации	строка	Доступно для виджетов визуализации.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`rql`	да (для виджетов визуализации)		Запрос для построения визуализации	RQL-запрос	Доступно для виджетов визуализации. Должно содержать конструкцию вида: `n =%{{n}}%` Здесь поле `n` соответствует `id` записи сущности домена.
`arguments`	да (для виджетов визуализации)		Атрибуты сущности, используемые в качестве переменных в RQL-запросе.	массив	Доступно для виджетов визуализации. Содержит пары полей `key` и `type`: • `key` — поле сущности домена, значение которого будет передаваться в переменную `%{{n}}%`. • `type` — тип поля, указанного в `key` (например, `string`, `datetime`)
`last`	да (для виджетов визуализации)		Временной период отображения данных в виджете.	объект	Доступно для виджетов визуализации. См. описание поля `last`.
`limit`	да (для виджетов визуализации)		Ограничение RQL-запроса на количество строк в результате.	целое положительное число	Доступно для виджетов визуализации.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`source`	да (для виджетов визуализации)		Тип источника данных (виджета) для построения визуализации.	перечисление	Доступно для виджетов визуализации. Устанавливается значение `search` — поиск по хранилищам и активным спискам.
`chartType`	да (для виджетов визуализации)		Тип визуализации.	перечисление	Доступно для виджетов визуализации. Доступные значения: • `BAR_CHART` — столбчатая диаграмма; • `LINE_CHART` — график; • `PIE_CHART` — круговая диаграмма; • `STACKED_BAR_CHART` — столбчатая диаграмма с накоплением; • `TABLE` — таблица; • `TOTAL` — итог.
`searchEntity`	да (для виджетов визуализации)		Имя переменной, которая соответствует хранилищу событий.	массив	Доступно для виджетов визуализации. См. описание поля `searchEntity`.
`nameMapping`	да (для виджетов визуализации)		Сопоставление полей данных с отображаемыми элементами.	массив	Доступно для виджетов визуализации. См. описание поля `nameMapping`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`eventStorageId`	да (для виджетов истории изменений)		Идентификатор хранилища событий.	строка	Ссылка на хранилище событий, из которого загружаются события для отображения в виджете.
`rules`	да (для виджетов списка связей)		Идентификатор правила определения набора связей.	объект	Правило определяет, какие связи могут быть доступны для просмотра и редактирования в виджете. Если поле не заполнено, будут доступны все связи сущности, в описание которой добавлен этот виджет. См. описание `rules`.
`icon`	нет (для виджетов дашбордов)		Ссылка на идентификатор медиаресурса.	View ID Ref	Иконка действия, связанного с виджетом.

▼ Схема записи виджета атрибута

```
widgets:
# Идентификатор виджета.
- id:
     # Тип виджета атрибута.
     type: attribute
     # Идентификатор атрибута (должен быть предварительно объявлен).
     attribute:
     # Настройки типа данных для виджета.
     control:
        # Элемент управления.
        type:
        # Настройки отображения и стиля.
     # Описание виджета.
     description:
```

▼ **Схема записи виджета системного атрибута**

```
widgets:
# Идентификатор виджета.
- id:
     # Тип виджета атрибута.
     type: attribute
     # Идентификатор атрибута (его не нужно объявлять предварительно).
     attribute:
     # Описание виджета.
     description:
```

▼ **Схема записи виджета действия над объектом**

```
widget:
# Идентификатор виджета.
- id:
     # Тип действия виджета действия.
     type: action
     # Ссылка на действие.
     action:
     # Элемент управления виджетом.
     control:
         # Элемент управления виджетом (стандартная кнопка).
         type: regular_button
         # Ссылка на идентификатор медиаресурса для иконки виджета.
         icon:
     # Описание виджета.
     description:
```

▼ **Схема записи виджета связи сущностей (компактный редактор)**

```
widgets:
# Идентификатор виджета.
- id:
  # Тип виджета связи.
  type: linkage
  # Ссылка на связь.
  linkage:
  # Элемент управления виджетом.
  label:
```

▼ **Схема записи виджета истории изменений**

```
widgets:
# Идентификатор виджета
- id:
  # Тип виджета истории изменений
  type: history
```

```
  # Отображаемое название виджета
  label:
  # Описание виджета
  description:
  # Ссылка на хранилище событий
  eventStorageId:
```

▼ Схема записи виджета комментариев

```
widgets:
# Идентификатор виджета
- id:
  # Тип виджета комментариев
  type: comments
  # Отображаемое название виджета
  label:
  # Описание виджета
  description:
```

▼ Схема записи виджета комментариев

```
widgets:
# Идентификатор виджета
- id:
  # Тип виджета списка связей
  type:
  # Отображаемое название виджета
  label:
  # Список объектов - записей правил доступности связей
  rules:
    # Список объектов - записей включения связей из определенных доменов
    includes:
      # ИД домена
      - id:
        # Массив ИД связей, которые будут включены в список (только они будут
включены)
        only:
        # Массив ИД связей, которые будут исключены из списка (а все прочие будут
включены)
        except:
    # Список объектов - записей исключения связей из определенных доменов
    excludes:
      # ИД домена
      - id:
        # Массив ИД связей, которые будут исключены из списка (только они будут
исключены)
        only:
        # Массив ИД связей, которые будут включены в список (а все прочие будут
исключены)
```

```
        except:
```

Структура `actions` (в виджетах действий)

Поле `actions` представляет собой массив ссылок на действия, связанные с сущностями в цепочке наследования — родителем и его потомками. Это позволяет описать в виджете действия родителя кнопку для действий, содержащую раскрывающийся список. Если ссылка на эту кнопку указана в интерфейсе списка объектов родителя, при нажатии на нее отображается список действий над объектами родителя и потомков, приведенный в поле `actions`. В списке можно выбрать объект требуемого потомка для выполнения указанного действия.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`action`	да		Ссылка на описание действия над объектом родителя или потомка.	View ID	

Структура `rules` (в виджетах списка связей)

Поле `rules` содержит правила доступности связей для просмотра и редактирования.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`includes`	нет		Правила включения связей.	массив объектов	Массив записей правил, определяющих, какие связи будут включены в список для отображения и редактирования. См. описание поля `includes`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`excludes`	нет		Правила исключения связей.	массив объектов	Массив записей правил, определяющих, какие связи не будут включены в список для отображения и редактирования. См. описание поля `excludes`.

Структура `includes` (в правилах доступности)

Поле `includes` содержит ссылку на домен и критерии доступности связей этого домена для просмотра и редактирования в рамках этого виджета.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Ссылка на идентификатор домена.	Data ID Ref или FQID	Домен, к которому применяется правило включения.
`only`	нет		Список идентификаторов связей (`linkage`) домена, доступных для просмотра и редактирования.	массив строк	В список доступных связей включаются только перечисленные в этом поле связи домена.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
except	нет		Список идентификаторов связей (linkage) домена, недоступных для просмотра и редактирования.	массив строк	В список доступных связей включаются все имеющиеся связи, кроме перечисленных в этом поле.

Структура excludes (в правилах доступности)

Поле excludes содержит ссылку на домен и критерии недоступности связей этого домена для просмотра и редактирования в рамках этого виджета.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
id	да		Ссылка на идентификатор домена.	Data ID Ref или FQID	Домен, к которому применяется правило исключения.
only	нет		Список идентификаторов связей (linkage) домена, недоступных для просмотра и редактирования.	массив строк	Из списка доступных связей исключаются только перечисленные в этом поле связи домена.
except	нет		Список идентификаторов связей (linkage) домена, доступных для просмотра и редактирования.	массив строк	Из списка доступных связей исключаются все имеющиеся связи, кроме перечисленных в этом поле.

Структура `control` (в виджете)

В поле `control` задаются элементы управления виджетом.

Поле `control` не задается в схеме домена для виджетов действий, если в описании действия, на которое ссылается виджет, указаны значения полей:

- `type` = `bulk`;
- `operation` = `edit_attribute`.

В этом случае система автоматически подставляет для действия элемент управления `dropdown` — выпадающий список значений, которые можно присвоить атрибуту объектов. Ссылка на атрибут указывается в описании действия.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`type`	да		Тип элемента управления для виджета	перечисление	Для виджета атрибута можно явно задать элемент управления в зависимости от того, в каком элементе представления размещается виджет и будет ли он редактируемым. Для виджета действия всегда устанавливается значение `regular_button` (стандартная кнопка). Возможные значения для виджета связи: • `select` — для выбора связи типа 1-1: • `multi_select` — для выбора связи типа 1-N или N-N (вы находитесь на стороне 1); • `table` — для выбора связи типа 1-N или N-N (вы находитесь на стороне 1). См. описание поля `table`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`falseText`	да (для виджетов атрибутов)		Наименование для значения `false`	строка	Поле доступно только в виджетах атрибутов для элемента управления `bool_text`.
`trueText`	да (для виджетов атрибутов)		Наименование для значения `true`	строка	Поле доступно только в виджетах атрибутов для элемента управления `bool_text`.
`icon`	нет		Доступно для виджетов действий, ссылка на идентификатор медиаресурса для иконки.	строка	

Структура `table` (в редакторе связей)

Поле `table` описывается только для элемента управления типа `table`. В этом поле описываются настройки расширенного редактора связей.

Оно включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`addLinksView`	да		Ссылка на списочное представление сущности, объекты которой доступны для связывания с текущим объектом в расширенном редакторе.	объект	
`columns`	да		Массив записей с описанием столбцов таблицы.	массив	См. описание поля `columns`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`sortingBy`	нет		Ссылки на правила сортировки атрибутов сущности.	объект	Правила сортировки связей определяются в схеме домена. Если хотя бы одно правило сортировки не определено в схеме домена, система сортирует связи в таблице по времени поступившей о них информации за односекундный интервал и далее сортирует их по идентификатору связи. См. описание поля `sortingBy`.
`paging`	нет		Пагинация данных.	объект	См. описание поля `paging`.

Структура `columns` (в таблице виджета связи)

Поле `columns` содержит массив записей столбцов таблицы.

Столбец включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных
`id`	да		Идентификатор столбца таблицы. Он отображается в заголовке столбца атрибута в таблице.	массив

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных
`attribute`	да		Атрибут сущности или связи, используемый для отображения значения в столбце таблицы.	строка

Структура `last` (в виджетах визуализации)

Поле `last` содержит параметры времени для визуализации.

Поле	Описание	Тип данных	Комментарий
`unit`	Единица измерения периода формирования визуализации.	перечисление	Доступные значения: • `DAY` — день (24 часа); • `HOUR` — час; • `MINUTE` — минута; • `MONTH` — месяц (30 дней); • `YEAR` — год;
`value`	Количество единиц измерения.	целое положительное число	

Структура `searchEntity` (в виджетах визуализации)

Поле	Описание	Тип данных	Комментарий
`id`	Идентификатор записи источника (активного списка или хранилища событий).	строка	
`type`	Тип источника.	перечисление	Доступные значения: • `active_list` • `event_storage`

Структура `nameMapping` (в виджетах визуализации)

Поле `nameMapping` содержит массив объектов с парами значений `key–value`, которые зависят от типа визуализации. Поле может содержать несколько пар "ключ — значение".

Поле	Обязательное поле	Тип данных	Комментарий
`key`	да	строка	Если в поле `chartType` установлено значение `BAR_CHART`, `LINE_CHART` или `STACKED_BAR_CHART`, доступны значения: • `x` — ось x; • `y` — ось y; • `group_by` — группировка. Если `chartType` = `PIE_CHART`, доступны значения: • `label` — ось x; • `value` — ось y; Если `chartType` = `TOTAL`, устанавливается значение `value`. Если `chartType` = `TABLE`, указывается название столбца таблицы.
`value`	да	строка	Если в поле `chartType` установлено значение `BAR_CHART`, `LINE_CHART`, `STACKED_BAR_CHART` или `PIE_CHART`, указывается поле запроса, из которого берется значение для указанного ключа. Если `chartType` = `TOTAL`, указывается поле запроса, по которому считается общее значение. Если `chartType` = `TABLE`, указывается поле запроса, из которого берется значение для указанного столбца.

Структура `groups`

Поле `groups` представляет собой массив записей визуальных элементов, на которые ссылается представление.

Поле `groups` включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Уникальный идентификатор группы представления.	View ID	
`description`	нет	Пустая строка	Описание группы представления.	строка	
`type`	да		Тип группы.	перечисление	Доступные значения: • `block` — блок; • `tab` — вкладка; • `form` — форма создания или редактирования объекта. Блоки, вкладки и формы описаны ниже.
`layout`	нет		Настройки взаимного расположения компонентов в группе.	размещение компонентов группы представлений	См. описание поля `layout`. Доступно только для типа `block`.
`components`	да		Массив записей блоков, из которых состоит вкладка.	массив	См. описание поля `components`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`menu`	нет		Меню группы представления.	массив	Доступно только для типа `tab`
`label`	нет	Пустая строка	Наименование группы представления.	строка	Доступно только для типов: • `block`; • `tab`.

Структура `layout` (в группе представления)

Поле `layout` представляет собой массив записей настроек взаимного расположения элементов. Доступно только для типа `block`.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`direction`	да		Порядок отображения элементов в записи группы	строка	Направление размещения определяет, как будут располагаться компоненты внутри группы. Возможные значения: • `column` — вертикальное размещение. Используется по умолчанию, если элемент размещения не был указан для группы. • `row` — горизонтальное размещение.
`expandable`	нет	false	Возможность развернуть элемент	булево	Доступные значения: • `true` — элемент можно развернуть; • `false` — элемент нельзя развернуть.

Структура `components`

Поле `components` используется для перечисления компонентов элемента поля `groups`. Такими компонентами могут быть:

- блоки (для элемента `tab`);
- виджеты.

Поле `components` включает следующие поля:

Поле	Описание	Тип данных	Комментарий
`block`	Уникальный идентификатор блока.	строка	Блоки перечисляются только для типа `tab`
`widget`	Уникальный идентификатор виджета.	строка	
`showLabel`	Отображение название компонента.	булево	Доступные значения: • `true` — отображать название; • `false` — не отображать название.

Структура `views` (панели и модальные окна)

Конфигурация интерфейса объекта включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Уникальный идентификатор представления одного объекта.	View ID	
`description`	нет	Пустая строка	Описание элемента домена.	строка	

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`type`	да		Тип представления одного объекта.	строка	Определяет тип представления одного объекта, который влияет на поведение и возможности интерфейса с учетом контекста использования. Возможные значения: • `panel` — информационная панель, которая может отображать вкладки (группы представлений с типом `tab`); • `modal` — модальное окно, которое может отображать форму (группу представлений с типом `form`).
`label`	нет	Пустая строка	Название, отображаемое в интерфейсе.	строка	

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`menu`	нет		Действия, отображаемые в меню панели.	массив	Доступно только для представлений с типом `panel`. См. описание поля.
`tabs`	да		Вкладки, отображаемые в панели.	массив	См. описание поля.
`form`			Ссылка на группу с типом `form`.	ссылка на форму	
`buttons`			Массив записей кнопок, отображаемых в модальном окне.	массив	Доступно только для представлений с типом `modal`. См. описание поля `buttons`.

▼ Схема записи панели

```
views:
     # Уникальный идентификатор.
     - id:
       # Описание.
       description:
       # Тип представления (панель).
       type: panel
       # Название панели.
       label:
      # Меню панели.
      menu:
         # Действие.
         - action:
           # Параметры управления.
           control:
             # Тип управления.
             type:
                # Идентификатор медиаресурса для кнопки.
                icon:
     # Массив записей вкладок панели.
     tabs:
         # Идентификатор вкладки.
         - tab:
```

▼ **Схема записи модального окна**

```
views:
 # Уникальный идентификатор.
 id:
 # Описание.
 description:
 # Тип представления (модальное окно).
 type: modal
 # Ссылка на форму (описывается в поле groups с типом form).
 form:
 # Название модального окна.
 label:
```

Структура `menu`

Поле `menu` представляет собой массив записей элементов меню.

Запись включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`action`	да		Ссылка на действие, вызываемое в меню.	View ID Ref	Действие, которое будет вызвано при нажатии на элемент меню. Действия, которые работают с контекстом списка, могут быть недоступны, если в списке не выбраны нужные данные. Например, действие массового удаления объектов не будет работать, если в списке не выбран ни один объект.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`control`	нет		Элемент управления.	элемент управления	Система по умолчанию подставляет элемент `regular_button`, для которого можно опционально указать ссылку на медиа-ресурс — иконку (поле `icon`). См. описание поля `control`.

Структура `control` (в меню)

В поле `control` задаются элементы управления виджетом или кнопками.

Поле	Описание	Тип данных	Комментарий
`type`	Элемент управления.	строка	Виджет действия управляется с помощью кнопки. В этом поле для него всегда устанавливается значение `regular_button` (стандартная кнопка).
`icon`	Идентификатор медиаресурса для иконки.	строка	

Структура `buttons`

Поле `buttons` используется для создания массива записей кнопок, отображаемых в модальном окне.

Запись включает следующие поля:

Поле	Описание	Тип данных	Комментарий
`type`	Идентификатор модального окна.	строка	Доступные значения типа: • `submit` — при нажатии на кнопку данные отправляются в систему; • `close` — при нажатии на кнопку окно закрывается, а данные не отправляются. Содержит поле `control`.
`label`	Название кнопки.	строка	Необязательное поле. Если значение не указано, система добавляет название по умолчанию.

▼ Схема записи кнопки

```
buttons:
    # Название кнопки.
    label:
     # Описание типа кнопки.
     - type:
        # Описание настроек параметров управления.
        control:
            # Тип параметра управления (кнопка).
            type: regular_button
            # Идентификатор медиаресурса для иконки кнопки.
            icon:
```

Структура `layout` (в интерфейсе дашбордов)

В поле `layout` задаются настройки взаимного расположения компонентов в интерфейсе.

Оно включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`type`	нет	grid	Тип взаимного расположения виджетов в интерфейсе.	перечисление	Значение `grid` позволяет расположить виджеты в виде сетки
`columns`	нет	3	Количество столбцов в интерфейсе виджетов.	число	Значение указывается, если в поле `type` указано значение `grid`.

Структура `table` (список объектов)

Поле `table` содержит настройки табличного отображения сведений о списке объектов.

Оно включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`columns`	да		Массив записей с описанием столбцов таблицы.	массив столбцов таблицы	См. описание поля `columns`.
`filter`	нет		Настройки фильтрации столбцов.	объект	См. описание поля `filter`.
`sortingBy`	нет		Ссылки на правила сортировки атрибутов сущности.	объект	См. описание поля `sortingBy`.
`paging`	нет		Пагинация данных.	объект	См. описание поля `paging`.
`refresh`	нет		Обновление данных.	объект	См. описание поля `refresh`.
`selection`	нет	`false`	Возможность выбора строки таблицы.	булево	См. описание поля `selection`.
`actions`	нет	Пустой массив	Действие при нажатии на строку таблицы.	массив действий на события таблицы	См. описание поля `actions`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`empty`	нет		Информация, отображаемая в таблице, если данные отсутствуют.	сообщение при отсутствии строк в таблице	См. описание поля `empty`.

Структура `columns` (в таблице списка объектов)

Поле `columns` содержит массив записей столбцов таблицы.

Столбец включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`id`	да		Идентификатор столбца таблицы.	массив	
`attribute`	да		Атрибут столбца.	строка	Можно использовать как обычные, так и системные атрибуты.
`join`	нет		Ссылка на идентификатор связи с сущностью, атрибут которой будет отображаться.	Data ID Ref	Значение в этом поле указывается, только если необходимо описать атрибут связанной сущности. Связь, на которую вы ссылаетесь, должна быть предварительно задана в поле `joins` в описании интерфейса списка объектов

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`type`			Тип редактора, используемый в столбце, в соответствии с таблицей.	строка	В этом поле можно изменить используемый по умолчанию тип редактора в столбце на другой допустимый тип. Допустимые типы для разных контекстов представлены в таблице.
`control`			Значения строк.	массив	Допустимо только для столбца типа "булево". В полях `trueText` и `falseText` указываются отображаемые строки для значений `True` и `False`: • `trueText: <строка>` • `falseText: <строка>`
`layout`	нет		Массив настроек размера и видимости столбца.	массив	См. описание поля `layout`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`label`	нет		Наименование столбца.	строка	Если значение этого поля не задано, в качестве наименования берется `id` столбца.

Структура `layout` (в столбце списка объектов)

Поле `layout` содержит массив настроек размера и видимости.

Настройка включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`visible`	нет	`true`	Видимость столбца.	строка	В этом поле доступны следующие значения: • `true` — столбец отображается (значение по умолчанию). • `false` — столбец скрыт.
`width`	нет		Ширина столбца в пикселях.	объект	См. описание поля `width`.

▼ Схема использования столбца

```
table:
  columns:
    # Идентификатор столбца.
    - id:
      # Атрибут столбца.
      attribute:
      # Тип данных в столбце.
      type
      # Массив настроек столбца.
```

```
      layout:
        # Параметр видимости столбца.
        visible:
        # Ширина столбца (по умолчанию, минимальная, максимальная).
        width:
          default:
          min:
          max:
```

Настройки ширины столбца (`width`)

Поле `width` содержит массив настроек ширины столбца.

Настройка включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных
`default`	да		Ширина столбца по умолчанию, в пикселях.	целое положительное число
`min`	да		Максимальная ширина столбца в пикселях.	целое положительное число
`max`	да		Минимальная ширина столбца в пикселях.	целое положительное число

Структура `filter`

Данные, по которым осуществляется фильтрация, должны быть описаны в настройках фильтрации. Фильтрация по этим данным в системе доступна независимо от того, отображаются ли они в интерфейсе списка объектов (таблице).

Фильтровать данные можно:

- по атрибутам сущности;
- по лейблу и атрибутам связанного объекта.

Фильтрация по атрибутам связанного объекта типа `Reference` не поддерживается.

Для ссылки на атрибут в настройках фильтрации используйте значение поля `id` в **описании атрибута** сущности.

Поле `filter` содержит массив настроек фильтрации столбцов таблицы.

Фильтр включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`attributes`	нет		Массив ссылок на атрибуты сущности, по которым производится фильтрация.		См. описание поля `attributes`.
`linkages`	нет		Массив ссылок на связи сущности, по которым производится фильтрация.		См. описание поля `linkages`.

Структура `attributes` (в фильтре списка объектов)

Поле `attributes` содержит массив ссылок на атрибуты сущности, по которым производится фильтрация.

Оно включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`type`	да		Тип фильтрации		В этом поле можно указать следующие типы фильтрации: • `specific` — фильтрация осуществляется по атрибутам сущности, ссылки на которые приведены в поле `use`. • `all` — фильтрация осуществляется по всем атрибутам. При использовании этого типа ссылки на атрибуты в поле `use` приводить не нужно.
`use`	да		Массив ссылок на атрибуты, по которым производится фильтрация		Задается, если в поле `type` указано значение `specific`. В этом поле перечисляются атрибуты, по которым осуществляется фильтрация в таблице, см. схему.

Структура `linkages` (в фильтре списка объектов)

Поле `linkages` используется:

- для задания ссылок на связи сущности, по которым производится фильтрация;
- для задания атрибутов связанных сущностей, по которым производится фильтрация.

Структура включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`type`	да (при ссылках на связи)		Тип фильтрации.		Заполняется при задании ссылок на связи сущности. Можно указать следующие типы фильтрации: • `specific` — фильтрация осуществляется по связям, ссылки на которые приведены в поле `use`. • `all` — фильтрация осуществляется по всем связям. При использовании этого типа ссылки на связи в поле `use` приводить не нужно.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
use	да (при ссылках на связи)		Массив ссылок на связи, по которым производится фильтрация.		Задается, если в поле type указано значение specific. В этом поле перечисляются связи, по которым осуществляется фильтрация в таблице, см. схему.
linkage	да (при ссылках на атрибуты)		Ссылка на связь с другой сущностью.		Заполняется при задании ссылок на атрибуты связанных сущностей. Поле linkedEntity этой связи содержит атрибуты связанной сущности, по которым будет доступна фильтрация.

▼ Схема фильтрации по отдельным атрибутам и связям

```
table:
  # Поле фильтрации.
  filter:
    # Тип фильтрации по отдельным атрибутам, перечисленным в поле use.
    attributes:
      type: specific
      use:
        - attribute: <attribute_1>
        - attribute: <attribute_2>
        ...
    # Тип фильтрации по отдельным связям, перечисленным в поле use.
    linkages:
      type: specific
      use:
```

```
          # Ссылка на связь с сущностью.
          - linkage: <entity_1>
            linkedEntity:
              # Ссылки на атрибуты связанной сущности, по которым будет
производиться фильтрация.
              - attribute: <attribute_3>
              - attribute: <attribute_4>
              ...
          - linkage: <entity_2>
          ...
```

Сортировка в столбцах таблицы (`sortingBy`)

Поле `sortingBy` содержит описание сортировки в столбцах таблицы.

Настройки включают следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`columns`	да		Массив записей столбцов, для которых возможна сортировка в таблице.	массив привязок правил сортировки к столбцам	В этом поле указывается: • в поле `column` — столбец, по которому выполняется сортировка; • в поле `sorting` — правило, в соответствии с которым она выполняется.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`defaults`	нет		Массив записей столбцов с указанием направления сортировки по умолчанию.	сортировка таблицы по умолчанию	В этом поле указывается: • в поле `column` — столбец для направления сортировки; • в поле `direction` — направление сортировки: ◦ `asc` — сортировка по возрастанию; ◦ `desc` — сортировка по убыванию.

Настройки количества записей в таблице (`paging`)

Поле `paging` содержит настройки количества записей, показываемых в таблице.

Настройка включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`sizes`	да		Количество записей, показываемых в таблице.	массив целых положительных чисел	В этом поле указываются варианты количества записей, доступные для выбора в настройке. Например, `[10, 50, 100]`.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`defaultSize`	да		Количество записей, показываемых по умолчанию.	целое положительное число	

Структура `refresh`

Поле `refresh` содержит настройки обновления записей, показываемых в таблице.

Настройка включает следующие поля:

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`mode`	да		Режимы обновления.	перечисление	В этом поле указываются режимы обновления записей: • `manual` — обновление производится вручную; • `auto` — обновление производится автоматически с заданным интервалом; • `all` — обновление производится вручную и автоматически.

Поле	Обязательное поле	Значение по умолчанию	Описание	Тип данных	Комментарий
`defaults`	да		Настройки автообновления.	объект	Поле доступно только для настроек фильтров таблиц с типами: • `all`; • `auto`. См. описание поля `defaults`.

Структура `defaults`

Поле `defaults` содержит настройки автообновления записей, показываемых в таблице. Используется, если в поле `mode` установлено значение `auto` или `all`.

Настройка включает следующие поля:

Поле	Описание	Тип данных	Комментарий
`enabled`	Включение и выключение автообновления.	булево	
`unit`	Единица измерения интервала автообновления.	перечисление	Доступные значения: • `second` — секунды; • `minute` — минуты; • `hour` — часы.
`interval`	Количество единиц измерения интервала автообновления.	целое положительное число	

Структура `selection`

Поле `selection` содержит настройки возможности выбора записей, показываемых в таблице.

Настройка включает следующие поля:

Поле	Описание	Тип данных	Комментарий
`selection`	Возможность выбора записей.	булево	Доступные значения: • `true` — выбор записей возможен; • `false` — выбор записей невозможен (значение по умолчанию).

Структура `actions` (в списке объектов)

Поле `actions` содержит настройки действий в таблице.

Настройка включает следующие поля:

Поле	Описание	Тип данных	Комментарий
`type`	Тип действия.	строка	Устанавливается значение `row_click` — действие при нажатии на строку.
`action`	Ссылка на идентификатор действия.	строка	

Структура `empty`

Поле `empty` содержит настройки отображения сообщения в таблице, если данные отсутствуют.

Настройка включает следующие поля:

Поле	Описание	Тип данных	Комментарий
`message`	Отображение сообщения в таблице, если данные отсутствуют.	строка	Необходимо указать отображаемое сообщение в поле `message`.

Типы данных в схемах доменов

Раздел содержит описания типов данных, поддерживаемых схемами доменов, с особенностями их использования и примерами.

Тип данных	Описание	Формат ввода-вывода	Пример
`UUID`	Универсальный уникальный идентификатор (UUID). Значение состоит из пяти групп шестнадцатеричных символов, разделенных дефисами: `<8 символов>-<4 символа>-<4 символа>-<4 символа>-<12 символов>` Используются только строчные буквы.	JSON-строка	`"61f0c404-5cb3-11e7-907b-a6006ad3dba0"`
`Timestamp`	Момент времени. В формате JSON-число не включает часовой пояс. В формате JSON-строка может включать часовой пояс; при сохранении всегда приводится к времени UTC.	JSON-число, JSON-строка	JSON-число: `1621428394335` JSON-строка: `"2024-08-10T10:31:41Z"`

Тип данных	Описание	Формат ввода-вывода	Пример

`Date`	Календарная дата. Ниже представлены форматы значения.
	
	Только дата:
	
	`YYYY`
	
	`YYYY-MM`
	
	`YYYY-MM-DD`
	
	Допустимы также формы "дата и время", состоящие из одной из вышеуказанных форм типа "только дата", за которой сразу же следует "T", затем одна из нижеследующих форм представления времени, а за ней — необязательное значение смещения на часовой пояс:
	
	`THH:mm`
	
	`THH:mm:ss`
	
	`THH:mm:ss.sss`
		JSON-строка	`"2024-05-23"`

Тип данных	Описание	Формат ввода-вывода	Пример

`Time`	Время. Ниже представлены форматы значения.
	
	Только дата:
	
	`YYYY`
	
	`YYYY-MM`
	
	`YYYY-MM-DD`
	
	Допустимы также формы "дата и время", состоящие из одной из вышеуказанных форм типа "только дата", за которой сразу же следует "T", затем одна из нижеследующих форм представления времени, а за ней — необязательное значение смещения на часовой пояс:
	
	`THH:mm`
	
	`THH:mm:ss`
	
	`THH:mm:ss.sss`
		JSON-строка	`"05:35:22"`

Тип данных	Описание	Формат ввода-вывода	Пример

`TimestampRange`	Отрезок момента времени.
	
	При сохранении всегда приводится к времени UTC.
	
	Ниже представлены форматы значения даты-времени начала и окончания.
	
	Только дата:
	
	`YYYY`
	
	`YYYY-MM`
	
	`YYYY-MM-DD`
	
	Допустимы также формы "дата и время", состоящие из одной из вышеуказанных форм типа "только дата", за которой сразу же следует "T", а затем одна из нижеследующих форм представления времени, а за ней — необязательное значение смещения на часовой пояс:
	
	`THH:mm`
	
	`THH:mm:ss`
	
	`THH:mm:ss.sss`
		JSON-строка	`["2024-08-10T10:31:41Z", "2025-10-10T10:31:41Z"]`

`String`	Строка. Задается в двойных кавычках.	JSON-строка	`"абвг123"`

Тип данных	Описание	Формат ввода-вывода	Пример
`MAC`	MAC-адрес. Значение — шесть двузначных шестнадцатеричных чисел, разделенных двоеточиями. Используются только заглавные буквы: `XX:XX:XX:XX:XX:XX`	JSON-строка	`"00:53:00:B8:DF:B8"`
`Integer`	Целое число	JSON-число	`255`
`Decimal`	Число с заданной точностью. Дробная часть числа отделяется точкой. По умолчанию используется степень точности с одной цифрой после десятичного разделителя. Если в параметрах указано количество знаков после запятой, используется это значение.	JSON-строка	`0.9`
`Float`	Число с плавающей точкой. Дробная часть числа отделяется точкой. По умолчанию используется степень точности с одной цифрой после десятичного разделителя. Если в параметрах указано количество знаков после запятой, используется это значение.	JSON-число	`0.999`

Тип данных	Описание	Формат ввода-вывода	Пример

`Bool`	Логическое значение	JSON-булево	`True`
			
			`False`

`Enum`	Список предопределенных значений с возможностью выбрать одно значение. Для того, чтобы использовать в схеме домена список, необходимо создать пользовательский тип данных.	JSON-строка	`"CRITICAL"`
`IpAddress`	Тип данных позволяет хранить IPv6 (по стандарту RFC 5952), IPv4, IPv4 в формате IPv6. IPv4 в нотации IPv6 записывается следующим образом: • Последние 32 младших бита записываются как IPv4-адрес. • В начале идут только нулевые поля и одно единичное, записанные в нотации IPv6.	JSON-строка	IPv6: `"2001:0db8:85a3:0000:0000:8a2e:0370:7334"` IPv4: `"192.0.2.128"` IPv4 в формате IPv6: `"::ffff:192.0.2.128"` (аналогично 192.0.2.128)
`CIDR`	Адрес подсети с маской в формате CIDR. Формат значения: `<IP-адрес>/<маска>`	JSON-строка	IPv4: `"198.51.100.0/22"` IPv6: `"2001:db8::/48"`

Тип данных	Описание	Формат ввода-вывода	Пример
`Reference`	Значением является ссылка (например, на сущность)	Ввод: JSON-строка, JSON-объект Вывод: JSON-объект	Ввод (JSON-строка): `"61f0c404-5cb3-11e7-907b-a6006ad3dba0"` Ввод (JSON-объект с обязательным полем id): `\{"id": "61f0c404-5cb3-11e7-907b-a6006ad3dba0"}` Вывод (JSON-объект): `\{"domainId": "exmpl.tld", "entityId": "Person", "id": "61f0c404-5cb3-11e7-907b-a6006ad3dba0"}`
`Workflow`	Рабочий процесс. Значением является список состояний и переходов. В базе хранится одно значение для сущности на текущий момент. Для того чтобы использовать в схеме домена рабочий процесс, необходимо создать пользовательский тип данных.	JSON-строка	`"REPAIRING"`
`Array`	Массив значений заданного типа данных. Элементы массива указываются в квадратных скобках и разделяются запятой и пробелом.	JSON-массив	`[321, 123, 12344]`
`Table`	Данные хранятся в виде массива JSON, содержащего объекты с системными полями и полем значений атрибутов. Системное поле — rowId. Значение rowId не должно повторяться.	JSON-массив	

Тип данных	Описание	Формат ввода-вывода	Пример
`Calculation`	Результат расчета по схеме. Число с плавающей точкой.	JSON-число	`100`

Скрипты

В данном разделе представлен обзор структуры, принципов работы, процесса создания и импорта скриптов в качестве зависимостей. Управление скриптами в системе осуществляется в разделе **Экспертиза**.

О скриптах

Скрипты — это сценарии, определяющие последовательность выполнения различных действий в системе.

Работа со скриптом

Доступные операции над скриптом:

- Создание скрипта
- Запуск скрипта
- Изменение скрипта
- Удаление скрипта

Создание скрипта

Скрипт можно создать самостоятельно или импортировать в систему.

Чтобы создать скрипт:

1. Перейдите в раздел **Экспертиза**. Система отобразит список элементов экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Скрипт**. Система отобразит окно создания скрипта.
3. Заполните поля скрипта, чтобы определить логику его работы.
4. Нажмите на кнопку **Опубликовать версию**, чтобы сохранить изменения и опубликовать созданный скрипт. Система отобразит уведомление об успешном создании скрипта. Новый скрипт отобразится в таблице раздела **Экспертиза**.

   Чтобы скрипт стал доступен для запуска, его необходимо включить.

   Вы можете создать скрипт без его публикации (в виде черновика) с помощью кнопки **Сохранить черновик**. Система отобразит уведомление об успешном создании черновика скрипта. Черновик отобразится в таблице раздела **Экспертиза**.

Запуск скрипта

После того как скрипт протестирован и сохранен, он готов к запуску.

Чтобы запустить скрипт:

1. Перейдите в раздел **Экспертиза**. Система отобразит список элементов экспертизы.
2. Нажмите на строку скрипта в списке. Система отобразит в правой части экрана карточку этого скрипта с подробной информацией о нем.
3. Убедитесь, что скрипт включен. Переключатель в карточке скрипта должен быть переведен в активное положение.
4. Нажмите на кнопку **Запустить** в карточке скрипта. Скрипт будет запущен.

Скрипт можно также запустить из раздела **Запуски**.

Процесс выполнения скрипта в системе

1. Если у скрипта есть аргументы, после запуска скрипта отображается окно для ввода аргументов. Типы аргументов совпадают с типами переменных и зависят от языка скрипта.
2. После задания аргументов система преобразует их в формат JSON, и передает скрипт на выполнение в воркер. При запуске скрипта система назначает ему специальный тег воркера. Нельзя путать этот тег с тегами, используемыми в **Экспертизе** для группировки скриптов по бизнес-модели. Тег воркера указывает, какой конкретный воркер будет выполнять этот скрипт.

   С помощью менеджера воркеров можно назначать определенным воркерам скрипты для выполнения, например, по языкам, на которых они написаны, или по компонентам системы.

   - Если аргументы в скрипте отсутствуют, в воркер передается пустой JSON-файл.
   - Если имеются непереопределенные аргументы по умолчанию, аргументы будут подставлены в JSON-файл.
   - Если имеются переопределенные аргументы по умолчанию, именно они будут подставлены в JSON-файл.
3. После получения скрипта воркер находит исходный код скрипта в сервисе экспертизы по идентификатору скрипта, анализирует его и преобразует в готовый для выполнения код.
4. Если необходимо отдельно установить определенные зависимости скрипта, воркер производит установку зависимостей в рамках отдельной задачи. Для этого воркер скачивает зависимости из реестра зависимостей, установленного по умолчанию. Установленные зависимости в дальнейшем хранятся на этом воркере. Они используются при последующих выполнениях скрипта на этом воркере, что существенно сокращает время работы.

   Импортируемые зависимости скрипта и сам скрипт должны быть написаны на одном языке.

   Если скрипту назначить другой тег воркера, он, возможно, будет передан на выполнение в другой воркер. Это повлечет за собой затрату дополнительного времени на установку зависимостей на этом воркере, так как ранее они не были на нем установлены.

5. После проверки наличия и установки зависимостей воркер проверяет, запускался ли этот скрипт ранее с такими же параметрами и сохранились ли результаты этого запуска в кэше.

Если такие данные сохранены в кэше, воркер предоставит их, что ускорит выполнение процесса.

▼ *Пример*

Скрипт производит определенную трансформацию входных данных в выходные данные — принимает объемные входные данные, преобразует их в JSON-файл, ищет определенное поле и передает его на выход.

Если такой скрипт с такими параметрами уже ранее запускался, воркер попытается найти кэшированные данные и предоставить их, не тратя времени на запуск скрипта.

6. Если готовый результат не найден, задача передается на выполнение. Каждый воркер в отдельный момент времени выполняет только одну задачу.
7. Воркер запускает процесс с интерпретатором требуемого языка. В файловой системе в отдельной директории сохраняются все необходимые данные. Например, аргументы, ссылки на зависимости, необходимые для запуска и указанные пользователем переменные окружения.
8. Система помещает скрипт в специальную оболочку, позволяющую через библиотеку **playbooks_sdk** бесшовно получать доступ к переменным, элементам экспертизы, запускам и интеграциям с другими сервисами.
9. Скрипт выполняется. Становятся доступными результат выполнения скрипта и логи. В зависимости от результата выполнения скрипта статус его запуска изменяется на **Выполнен** или **Ошибка**.
10. Воркер завершает процесс, кэширует результаты и удаляет директорию с артефактами работы, так как все необходимые данные уже сохранены. После этого воркер переходит в режим ожидания следующей задачи.

Изменение скрипта

Чтобы изменить скрипт:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку скрипта в списке. Система отобразит в правой части экрана карточку с подробной информацией о скрипте.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек скрипта.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

4. Внесите требуемые изменения в конфигурацию скрипта.
5. Нажмите на кнопку **Обновить версию**. Новая версия скрипта с измененной конфигурацией будет опубликована. Система отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию скрипта в виде черновика без публикации помощью кнопки **Сохранить черновик**. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки скрипта будет отображаться предупреждение о наличии неопубликованных изменений.

Удаление скрипта

Чтобы удалить скрипт:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку скрипта в списке. Система отобразит в правой части экрана карточку с подробной информацией о скрипте.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления плейбука.
4. Нажмите на кнопку **Удалить**. Система удалит скрипт и отобразит соответствующее уведомление. Удаленный скрипт будет исключен из списка элементов экспертизы.

Вы также можете удалить группу элементов экспертизы. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку 🗑 на панели инструментов. Отобразится окно удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные элементы и отобразит соответствующее уведомление. Удаленные элементы будет исключены из списка элементов экспертизы.

Структура RObject-схемы скрипта и информация об API

Поля метаданных

Запись метаданных включает следующие поля:

Поле	Описание	Тип данных
`id`	Уникальный идентификатор скрипта	строка
`type`	Тип элемента экспертизы	В скрипте здесь всегда устанавливается значение `script`.
`name`	Название. Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`), точка, подчеркивание и дефис без пробелов.	строка

Поле	Описание	Тип данных
`version`	Версия скрипта в формате Semantic Versioning	строка

▼ *Пример заполненной структуры метаданных:*

```
id: script_ new_version
type: script
name: script_ new_version_name
version: 1.0.0
```

Поле `source`

В первой строке блока в поле `source` указывается пользовательский тег, определяющий язык, на котором написан код скрипта.

Допустимые значения тега в поле `source`:

Тег	Язык скрипта
`!bun`	TypeScript, среда выполнения Bun
`!deno`	TypeScript, среда выполнения Deno
`!py`	Python
`!go`	Go
`!pwsh`	PowerShell
`!bash`	Bash

Начиная со второй строки, в блоке указывается исходный код скрипта.

Для всех языков, кроме PowerShell и Bash, в блоке кода обязательно должна быть определена функция `main`, являющаяся входной точкой скрипта. Аргументы этой функции, таким образом, будут входными аргументами скрипта.

Для языков PowerShell и Bash аргументы задаются с помощью глобальных переменных.

Доступ к компонентам платформы

Из SDK можно получать доступ к компонентам платформы (например, управлению доменами).

Синтаксис импортируемых зависимостей

При импорте скриптов экспертизы в качестве зависимостей используйте следующий синтаксис для обращения к скрипту:

TypeScript (bun, deno)

- `@expertise/<source_id>` — будет импортироваться самая последняя версия скрипта;
- `@expertise/<source_id>@0.0.9` — будет импортироваться указанная версия скрипта, в данном случае, 0.0.9.

Здесь <source_id> — id элемента экспертизы типа **Скрипт**.

▼ *Пример*

```
id: bun_with_expertise_dependency
name: Bun with expertise dependency
version: 0.0.6
type: script
source: !bun |
  import { func } from '@expertise/bun_dependency_one_func@0.0.9'
  import { func2 } from '@expertise/bun_dependency_one_func_2'

  export async function main() {
    console.log('main f')
    func()
    console.log('main f2')
    func2()
  }
```

Задание значений аргументов в скрипте

При написании скрипта можно указать аргументы, которые скрипт будет принимать при запуске. У языков, где есть функция `main`, ее аргументы будут аргументами скрипта. У языков, где нет функции `main` (или другой точки входа), входные аргументы могут быть заданы через специальный синтаксис глобальных переменных.

После того как скрипт импортируется в систему, исходный код анализируется и проверяется на наличие входных аргументов. При запуске скрипта вручную, по расписанию, по триггеру или в качестве очередного этапа плейбука, могут быть заданы аргументы, если они есть.

Некоторые языки позволяют задавать значения для аргументов по умолчанию. В таком случае аргументы становятся опциональными.

Переменные окружения

Для скрипта можно задавать переменные окружения в следующем формате:

```
<KEY>=<value>
```

Здесь <KEY> — название переменной окружения, а <value> — значение переменной окружения.

Максимально допустимое время выполнения скрипта

Для скрипта можно задавать максимально допустимое время его выполнения (тайм-аут). По истечении этого времени процесс будет прерван принудительно.

Для задания тайм-аута скрипта используется переменная окружения `TIMEOUT`.

Доступ к хранилищу данных

Когда скрипт запускается, система создает в специальном хранилище данных запись о запуске. Эти записи можно использовать в скриптах и плейбуках, например, как условие срабатывания определенного правила.

Каждая запись в хранилище имеет свой идентификатор (ID интеграции), на который пользователь может ссылаться в скриптах.

Использование API в скриптах

API используется в системе для выполнения запросов в скриптах и манипуляций с данными в рамках системы.

Языки, поддерживаемые системной библиотекой

В системе имеется собственная библиотека **playbooks_sdk**. Ее можно использовать при написании скриптов на следующих языках:

- Python;
- TypeScript (среды выполнения Bun и Deno);
- Go;
- VRL.

Для языков Python и TypeScript вы можете указать версию библиотеки, которую необходимо импортировать. По умолчанию система всегда импортирует самую последнюю версию библиотеки.

Доступ к переменным и к секретам

Библиотека **playbooks_sdk** позволяет получать доступ к переменным и секретам в системе.

Использование переменных на примере языка Python:

```
import playbooks_sdk

playbooks_sdk.get_variable("u/user/foo")
playbooks_sdk.set_variable("u/user/foo", value)
```

Использование секретов на примере языка Python:

```
import playbooks_sdk

playbooks_sdk.get_secret("u/user/foo")
playbooks_sdk.set_secret("u/user/foo", value)
```

Работа с конфигурациями

Можно работать с уже добавленными в систему конфигурациями. Конфигурация представляет собой задаваемый пользователем набор полей, который используется для соединения с внешними системами и передачи данных в эти системы.

Поля конфигурации можно задавать в виде значений или указывать как ссылки на существующие в системе переменные или секреты.

В языках, поддерживаемых системной библиотекой **playbooks_sdk**, заданные поля конфигурации превращаются в конструкции языка.

Работа со схемами конфигураций

При написании скриптов на языке с сильной типизацией (например, TypeScript) можно получать схемы (типы) используемых конфигураций.

Поля настроек выполнения скрипта

Блок настроек выполнения скрипта является необязательным.

Если поля, указанные в нем, заполнены, они являются значениями по умолчанию для соответствующих параметров самой задачи.

Блок включает следующие поля:

Поле	Описание	Тип данных
`concurrent_limit`	Количество параллельных выполнений скрипта. Параметр ограничивает количество параллельных выполнений скрипта. Используется, если для корректной работы скрипта необходимо ввести это ограничение. Например, у сервера, к которому обращается задача при выполнении скрипта, может быть ограниченное количество входных соединений. Этот параметр позволяет ограничить количество выполняющихся экземпляров скрипта количеством входных соединений. В результате будет установлено разрешенное количество соединений для выполнений скрипта, а ограничение на сервере превышено не будет. Если количество параллельных выполнений не указано в самой задаче, система берет его значение из этого поля.	целое число

Поле	Описание	Тип данных
`concurrency_time_window_t`	Максимально допустимое время выполнения задачи при параллельном выполнении в секундах. Параметр ограничивает время выполнения задачи. Если оно превысит значение, указанное в этом поле, задача будет отменена.	целое число
`cache_ttl`	Время жизни (TTL, time to live) кэша данных скрипта в секундах. Параметр можно использовать при работе скрипта с внешним объектом (например, активными списками), чтобы соотносить время жизни кэша данных скрипта и объекта. Он позволяет не хранить данные в кэше скрипта дольше, чем они хранятся в кэше объекта. Время хранения определяется этим параметром. Если время жизни кэша данных скрипта не указано в самой задаче, система берет его значение из этого поля.	целое число

Составление скрипта

Скрипт создается в три этапа:

1. Описание блока метаданных.
2. Описание блока исходного кода скрипта на допустимом языке.
3. Описание блока настроек скрипта (опционально).

Метаданные скрипта

Заполните блок метаданных скрипта.

Код скрипта

После блока метаданных в скрипте размещается блок исходного кода скрипта. Этот код будет выполняться на воркере.

▼ *Пример кода (среда Bun)*

```
# Тег, определяющий язык, на котором написан код скрипта.
source: !bun
    # Функция main - входная точка скрипта.
    export async function main() {
      console.log("BUN SCRIPT EXECUTED SUCCESSFULLY")
      return 0
```

```
    }
```

Настройки выполнения скрипта

При необходимости заполните настройки в блоке настроек выполнения скрипта.

Пример 106. Пример настроек

```
id: script_ new_version
type: script
name: script_ new_version_name
version: 1.0.0

source: !bun
    export async function main() {
        console.log("BUN SCRIPT EXECUTED SUCCESSFULLY");
        return 0;
    }

concurrent_limit: 3
    concurrency_time_window_t: 1
    cache_ttl: 1
```

Пример скрипта

```
# Блок метаданных.
id: script_new_version
type: script
name: script_new_version_name
version: 1.0.0

# Блок кода скрипта.
source: !bun
    export async function main() {
      console.log("BUN SCRIPT EXECUTED SUCCESSFULLY")
      return 0
    }

# Блок настроек.
concurrent_limit: 3
    concurrency_time_window_t: 1
    cache_ttl: 1
```

Дополнительные настройки скрипта

Настройки повторного запуска скрипта

Максимальное количество попыток запуска скрипта в случае его выполнения с ошибкой и интервал

между попытками запуска скрипта задаются в параметрах триггера, в который добавлен скрипт.

Если скрипт запускается неоднократно и в нем используется конфигурация с указанной в ней переменной или секретом, результаты работы скрипта могут отличаться от запуска к запуску. Это может произойти, если в промежутке между запусками значение переменной или секрета было изменено.

API в скриптах

API предоставляет пользователям возможность работать программными средствами с объектами доменов. Для использования API при работе с объектами доменов используется пакет SDK `@playbooks/sdk`. Он содержит клиенты для работы со следующими ресурсами:

- объекты доменов (`cms`);
- активные списки (`activeList`);
- схемы расчетов (`calculation`);
- конфигурации (`ConfigurationClient`);
- переменные (`VariableClient`).

Вызовы API осуществляются через скрипт, написанный на языке TypeScript или Python.

Использование API для объектов доменов

Клиент, содержащийся в пакете, позволяет работать с объектами (экземплярами сущностей) доменов с использованием описанных ниже методов.

Методы API для работы с объектами доменов включают:

- Получение информации — извлечение данных об отдельных объектах и их связях, а также о списках объектов с фильтрацией по атрибутам и связям.
- Управление объектами — создание, полное обновление, частичное изменение и удаление объектов.

Описания и примеры приведены для TypeScript (среда выполнения Bun) и Python.

Подготовка клиента для работы с объектами

Чтобы работать с объектами доменов с помощью пакета SDK, импортируйте его:

Пример 107. Синтаксис импорта в TypeScript

```
import * as sdk from '@playbooks/sdk'
...
export async function main() {
...
  const v = await sdk.cms.<method_name>(<arg_1>, <arg_2>, ..., <arg_m>)
  console.log(v)
...
}
```

Пример 108. Синтаксис импорта в Python

```
from playbooks.cms import CmsClient, Filter, Sorting, Pagination, Order
...
def main() {
...
    client = CmsClient.initialize()
    v = client.<method_name>(<arg_1>, <arg_2>, ..., <arg_m>)
    print(v)
...
}
```

Здесь:

- `<method_name>` — имя используемого метода;
- `<arg_N>` — параметры метода.

Получение данных об объекте (`get`)

Метод используется для получения определенного объекта.

Пример 109. Синтаксис get в TypeScript

```
get({
  domainId: <domain_id>, // string
  entityId: <entity_id>, // string
  id: <object_id>, // string
  refFields: <reference_attributes>, // string[] | undefined
})
```

Пример 110. Синтаксис get в Python

```
get(
    domain_id=<domain_id>, # str
    entity_id=<entity_id>, # str
    id=<object_id>, # str
    ref_fields=<reference_fields> # List[str] | None
)
```

Параметры метода включают в себя:

- `<domain_id>` — идентификатор домена (строка);
- `<entity_id>` — идентификатор сущности (строка);
- `<object_id>` — идентификатор объекта (строка);
- `<reference_fields>` (необязательный) — дополнительные поля, которые нужно включить в ответ. Это поля объектов, на которые ссылаются атрибуты типа Reference (массив строк вида `refAttributeId.attributeId`).

Примеры получения данных об объекте

▼ *Пример скрипта с* `get` *в TypeScript*

```
id: get_object_in_cms_ts
name: 'get_object_in_cms_ts'
version: 0.0.1
type: script
tags: [tests, typescript, bun, cms]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  // Инициализация переменных domainId, entityId и objectId - id экземпляра
сущности.
  const domainId = 'example.com'
  const entityId = 'Person'
  const objectId = 'a1b2c3d4-5678-90ab-cdef-1234567890ab'

  export async function main() {
    try {
      // Вызов функции получения объекта с передачей objectId, domainId и entityId.
      const cmsObject = await sdk.cms.get({ domainId, entityId, id: objectId })

      // Вывод полученного объекта в консоль.
      console.log(cmsObject)
    } catch (error) {
      // Вывод ошибки в консоль, если произошла ошибка при получении объекта.
      console.log(error)
    }
  }
```

▼ *Пример скрипта с* `get` *в Python*

```
id: get_object_in_cms_py
name: get_object_in_cms_py
version: 0.0.1
type: script
tags: [tests, python, cms]
source: !py |
    from playbooks.cms import CmsClient, Filter, Sorting, Pagination, Order

    # Инициализация переменных domain_id, entity_id и object_id - id домена,
сущности и объекта.
    domain_id = 'example.com'
    entity_id = 'Person'
    object_id = 'a1b2c3d4-5678-90ab-cdef-1234567890ab'

    def main():
        try:
            client = CmsClient.initialize()
            cms_object = client.get(id=object_id, domain_id=domain_id,
```

```
entity_id=entity_id) # Получение объекта по указанному id.

            print('Print object:') # Вывод объекта в консоль.
            print(cms_object)

        except Exception as error:
            # Обработка и вывод ошибки, если произошла ошибка при получении объекта.
            print('An exception occurred:', error)
```

Получение данных о связях объекта (`find_object_links`)

Метод используется для получения массива связей объекта. Фильтры для уточнения запроса, порядок сортировки и количество выводимых результатов задаются с помощью параметров.

Пример 111. Синтаксис find_object_links в TypeScript

```
find_object_links({
  domainId: <domain_id>, // string
  entityId: <entity_id>, // string
  id: <object_id>, // string
  linkRequests: [
    {
      domainId: <domain_id>, // string
      linkageId: <linkage_id>, // string
      filterLink: {
        conditions: <conditions>, // string | undefined
        linkIds: <link_ids> // string[] | undefined
        objectIds: <object_ids> // string[] | undefined
      },
      side: <side_number>, // 1 | 2 | undefined
      sorting: {
        order: <order_dir>, // 'asc' | 'desc'
        orderBy: <order_rule>, // string
      },
      pagination: {
        limit: <limit>, // number
        offset: <offset>, // number | undefined
      },
      refFields: <reference_fields>, // string[] | undefined
    },
    ...,
  ],
})
```

Пример 112. Синтаксис find_object_links в Python

```
find_object_links(
    domain_id=<domain_id>, # str
    entity_id=<entity_id>, # str
    id=<object_id>, # str
```

```
    link_requests=[
        LinkRequest(
            domain_id=<domain_id>, # str
            linkage_id=<linkage_id>, # str
            filter_link=FilterLink(
                conditions=<conditions>, # str | None
                link_ids=<link_ids>, # List[str] | None
                object_ids=<object_ids> # List[str] | None
            ),
            side=<side_number>, # 1 | 2 | None
            sorting=Sorting(
                Order.<order_dir>, # Desc | Asc
                <order_rule> # str
            ),
            pagination=Pagination(
                <limit>, # int
                <offset> # int | None
            ),
            ref_fields=<reference_fields>, # List[str] | None
        ),
        ...,
    ]
)
```

Параметры метода включают в себя:

- `<domain_id>` — идентификатор домена (строка);
- `<entity_id>` — идентификатор сущности (строка);
- `<object_id>` — идентификатор объекта (строка);
- массив запросов для фильтрации связей объекта. Каждый запрос включает:
  - `<domain_id>` — идентификатор домена (строка);
  - `<linkage_id>` — идентификатор связи в домене (строка);
  - настройки фильтрации списка связанных объектов (необязательный блок):
    - `<conditions>` (необязательный) — условие фильтрации в виде строки, содержащей JSON в формате запросов MikroORM (строка);
    - `<link_ids>` (необязательный) — список идентификаторов связей (массив строк);
    - `<object_ids>` (необязательный) — список идентификаторов объектов (массив строк);
  - `<side_number>` (обязателен только при симметричной направленной связи) — сторона связи (число, `1` или `2`);
  - настройки сортировки списка по данным объектов на другой стороне связи (необязательный блок):
    - `<order_dir>` — порядок сортировки (`'asc'`/`Asc` — по возрастанию или `'desc'`/`Desc` — по убыванию);
    - `<order_rule>` — идентификатор одного из правил сортировки, определенного в поле

`sorting` для сущности на другой стороне связи (строка);

- ограничение количества выводимых результатов (необязательный блок):
  - `<limit>` — максимальное количество записей в ответе (число);
  - `<offset>` (необязательный) — порядковый номер первой записи, начиная с которой выводится ответ (число);
- `<reference_fields>` (необязательный) — дополнительные поля, которые нужно включить в ответ. Это поля объектов, на которые ссылаются атрибуты типа Reference сущности на другой стороне связи (массив строк вида `refAttributeId.attributeId`).

Примеры получения данных о связях объекта

В приведенных примерах скрипт запрашивает список связей типа `ProjectContribution` указанного объекта `Person`. У объектов на другой стороне связи значение поля `priority` должно быть равным `high`. Запрос возвращает результаты отсортированными в порядке возрастания согласно правилу `projectName`, указанному в объектах на другой стороне связи.

▼ *Пример скрипта с `find_object_links` в TypeScript*

```
id: find_object_links_ts
name: SDK CMS Find Object Links
version: 0.0.2
type: script
tags: [links]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  // Инициализация переменных domainId, entityId и objectId - id экземпляра
сущности.
  const domainId = 'example.com'
  const entityId = 'Person'
  const objId = '694aa0c4-70c9-4837-9beb-720535c44eb4'

  export async function main() {
    try {
      // Вызов функции получения связей объекта с передачей objId, domainId и
entityId.
      const findObjectLinks = await sdk.cms.find_object_links({
        domainId,
        entityId,
        id: objId,
        linkRequests: [
          {
            domainId: 'example.com',
            filterLink: {
              conditions: '{"$or":[{"priority":{"$eq":"high"}}]}',
            },
            linkageId: 'ProjectContribution',
            sorting: {
              order: 'asc',
```

```
            orderBy: 'projectName',
          },
        },
      ],
    })

    // Вывод полученных связей объекта в консоль.
    console.log(findObjectLinks)
  } catch (e) {
    // Вывод ошибки в консоль, если произошла ошибка при получении связей объекта.
    console.log(e)
  }
}
```

▼ *Пример скрипта с* `find_object_links` *в Python*

```
id: find_object_links_py
name: SDK CMS Find Object Links
version: 0.0.2
tags: [links]
type: script
source: !py |
    from playbooks.cms import FilterLink, LinkRequest, CmsClient, Sorting, Order

    domain_id = 'example.com'
    entity_id = 'Person'
    object_id = '694aa0c4-70c9-4837-9beb-720535c44eb4'

    def main():
        client = CmsClient.initialize()
        filter_link = FilterLink(
            conditions='{\"$or\":[{\"priority\":{\"$eq\":\"high\@}}]}'
        )

        sorting = Sorting(Order.Asc, 'projectName')

        link_request = LinkRequest(
            domain_id='example.com',
            linkage_id='ProjectContribution',
            filter_link=filter_link,
            sorting=sorting
        )

        res = client.find_object_links(id=object_id, domain_id=domain_id,
entity_id=entity_id, link_requests=[link_request])
        print(res[0].id)
        print(res[0].values)
```

Получение списка объектов (`list`)

Метод используется для получения отфильтрованного и отсортированного массива необходимых объектов. Фильтры для уточнения запроса, порядок сортировки и количество выводимых результатов задаются с помощью параметров.

Пример 113. Синтаксис list в TypeScript

```
list({
  domainId: <domain_id>, // string
  entityId: <entity_id>, // string
  filter: {
    conditions: <conditions>, // string | undefined
    ids: <object_ids>, // string[] | undefined
    query: <query>, // string | undefined
    links: {
      <operator>: [
        {
          domainId: <domain_id>, // string
          linkageId: <linkage_id>, // string
          ids: <link_ids>, // string[] | undefined
          object: {
            conditions: <links_conditions>, // string | undefined
            ids: <linked_object_ids> // string[] | undefined
          },
          side: <side_number> // 1 | 2 | undefined
        },
        ...,
      ]
    },
  },
  sorting: {
    order: <order_dir>, // 'asc' | 'desc'
    orderBy: <order_rule>, // string
  },
  pagination: {
    limit: <limit>, // number
    offset: <offset>, // number | undefined
  },
  refFields: <reference_fields>, // string[] | undefined
})
```

Пример 114. Синтаксис list в Python

```
list(
  domain_id=<domain_id>, # str
  entity_id=<entity_id>, # str
  filter=Filter(
      conditions=<conditions>, # str | None
      ids=<object_ids>, # str | None
      query=<query>, # str | None
```

```
        links=Links(
            operator=<operator>, # str
            links=[
                Link(
                    domain_id=<domain_id>, # str
                    linkage_id=<linkage_id>, # str
                    ids=<link_ids>, # List[str] | None
                    object=FilterLink(
                        conditions=<links_conditions>, # str | None
                        ids=<linked_object_ids>, # List[str] | None
                    ),
                    side=<side_number>, # 1 | 2 | None
                ),
                ...
            ]
        ),
    ),
    sorting=Sorting(
        Order.<order_dir>, # Desc | Asc
        <order_rule> # str
    ),
    pagination=Pagination(
        <limit>, # int
        <offset> # int | None
    ),
    ref_fields=<reference_fields>, # List[str] | None
)
```

Параметры метода включают в себя:

- `<domain_id>` — идентификатор домена (строка);
- `<entity_id>` — идентификатор сущности (строка);
- настройки фильтрации списка (необязательный блок):
  - `<conditions>` (необязательный) — условие фильтрации в виде строки, содержащей JSON в формате запросов MikroORM (строка);
  - `<object_ids>` (необязательный) — список идентификаторов требуемых объектов (массив строк);
  - `<query>` (необязательный) — запрос для поиска по атрибутам сущностей, на которые ссылается поле `label`, находящееся в поле `entities` схемы домена (строка);
  - запрос для фильтрации по связям объектов, который включает:
    - `<operator>` — логический оператор `and` (все перечисленные условия должны быть выполнены) или `or` (достаточно выполнения хотя бы одного условия из перечня);
    - `<domain_id>` — идентификатор домена (строка);
    - `<linkage_id>` — идентификатор связи в домене (строка);
    - `<link_ids>` (необязательный) — список идентификаторов связей (массив строк);

        - `<links_conditions>` (необязательный) — условие фильтрации в виде строки, содержащей JSON в формате запросов MikroORM (строка);
        - `<linked_object_ids>` (необязательный) — список идентификаторов связанных объектов (массив строк);
        - `<side_number>` (обязателен только при симметричной направленной связи) — сторона связи (число, `1` или `2`);
- настройки сортировки списка (необязательный блок):
    - `<order_dir>` — порядок сортировки (`'asc'`/`Asc` — по возрастанию или `'desc'`/`Desc` — по убыванию);
    - `<order_rule>` — идентификатор одного из правил сортировки, определенного в поле `sorting` сущности (строка);
- ограничение количества выводимых результатов (необязательный блок):
    - `<limit>` (обязателен, если ограничение указано) — максимальное количество записей в ответе (число);
    - `<offset>` (необязательный) — порядковый номер первой записи, начиная с которой выводится ответ (число);
- `<reference_fields>` (необязательный) — дополнительные поля, которые нужно включить в ответ. Это поля объектов, на которые ссылаются атрибуты типа Reference (массив строк вида `refAttributeId.attributeId`).

Примеры получения списка объектов

В приведенных примерах скрипт запрашивает список экземпляров сущности `Person`, которые имеют указанные идентификаторы объектов и атрибут `department` со значением `IT department`. Запрос должен вернуть не более 10 результатов, отсортированных согласно правилу `fullName` в порядке возрастания.

▼ *Пример скрипта с `list` в TypeScript*

```
id: list_object_in_cms_ts
name: 'list_object_in_cms_ts'
version: 0.0.1
type: script
source: !bun |
  import * as sdk from '@playbooks/sdk'

  // Инициализация переменных domainId и entityId.
  const domainId = 'example.com'
  const entityId = 'Person'

  export async function main() {
    try {
      // Вызов функции получения объекта с передачей domainId и entityId.
      const cmsObjects = await sdk.cms.list({
        domainId,
        entityId,
        filter: {
```

```
          conditions: '{\"$or\":[{\"department\":{\"$eq\":\"IT department\"}}]}',
          ids: [
            'a1b2c3d4-5678-90ab-cdef-1234567890ab',
            'f1e2d3c4-b567-89a0-bcde-0987654321ff',
            ...,
          ],
        },
        pagination: {
          limit: 10,
          offset: 0,
        },
        sorting: {
          order: 'asc',
          orderBy: 'fullName',
        },
      })

      // Вывод полученных объектов в консоль.
      console.log(cmsObjects)
    } catch (e) {
      // Вывод ошибки в консоль, если произошла ошибка при получении объектов.
      console.log(e)
    }
  }
```

▼ *Пример скрипта с* `list` *в Python*

```
id: list_object_in_cms_py
name: list_object_in_cms_py
version: 0.0.1
type: script
tags: [tests, python, cms]
source: !py |
    from playbooks.cms import CmsClient, Filter, Sorting, Pagination, Order

    # Инициализация переменных domain_id и entity_id - id домена и сущности.
    domain_id = 'example.com'
    entity_id = 'Person'

    def main():
        try:
            client = CmsClient.initialize()
            cmsObjects = client.list(
                domain_id=domain_id,
                entity_id=entity_id,
                filter=Filter(
                    '{\"$or\":[{\"department\":{\"$eq\":\"IT department\"}}]}',
                    [
                        'a1b2c3d4-5678-90ab-cdef-1234567890ab',
                        'f1e2d3c4-b567-89a0-bcde-0987654321ff',
```

```
                        ...,
                    ],
                    None),
                sorting=Sorting(Order.Asc, 'fullName'),
                pagination=Pagination(10, 0)
            )  # Получение списка объектов с указанными параметрами.

            # Вывод списка объектов в консоль.
            print('print objects:')
            for obj in cmsObjects:
                print(obj)

        except Exception as error:
            # Обработка и вывод ошибки, если произошла ошибка при получении списка
объектов.
            print('An exception occurred:', error)
```

Примеры фильтрации по связям

В примерах приведены значения поля `filter` для фильтрации полученных данных по связям.

Пример 115. Получение объектов, у которых нет ни одной указанной связи с любым другим объектом

```
{
  'links': {
    'and': [
      {
        'domainId': 'example.com',
        'linkageId': 'ProjectContribution',
        'object': {
          'conditions': '{\"$none\":{}}'
        },
        'side': 2
      }
    ]
  }
}
```

Пример 116. Получение списка объектов, у которых нет ни одной указанной связи с определенным объектом

```
{
  'links': {
    'and': [
      {
        'domainId': 'example.com',
        'linkageId': 'ProjectContribution',
        'object': {
          'conditions': '{\"$none\":{\"id\":\"9b2f8822-b932-44a7-991b-
3cb432cd6b98\"}}' // Условие фильтрации, здесь '9b2f8822-b932-44a7-991b-3cb432cd6b98'
-- идентификатор объекта, с которым у возвращенных объектов не должно быть связей.
```

```
        },
        'side': 2
      }
    ]
  }
}
```

Пример 117. Получение списка объектов, у которых есть хотя бы одна указанная связь с определенным объектом

```
{
  'links': {
    'and': [
      {
        'domainId': 'example.com',
        'linkageId': 'ProjectContribution',
        'object': {
          'conditions': '{\"$some\":{\"id\":\"9b2f8822-b932-44a7-991b-
3cb432cd6b98\"}}' // Условие фильтрации, здесь '9b2f8822-b932-44a7-991b-3cb432cd6b98'
-- идентификатор объекта, с которым у возвращаемых объектов должна быть хотя бы одна
связь.
        },
        'side': 2
      }
    ]
  }
}
```

Пример 118. Получение списка объектов, имеющих хотя бы одну указанную связь с объектами, в которых значение поля `name` равно 'Ivan'

```
{
  'links': {
    'and': [
      {
        'domainId': 'example.com',
        'linkageId': 'ProjectContribution',
        'object': {
          'conditions': '{\"$some\":{\"name\":\"Ivan\"}}'
        },
        'side': 2
      }
    ]
  }
}
```

Пример 119. Получение списка объектов, не имеющих ни одной указанной связи с объектами со значением поля `name`, равным 'Ivan'

```
{
  'links': {
    'and': [
      {
        'domainId': 'example.com',
        'linkageId': 'ProjectContribution',
        'object': {
          'conditions': '{\"$none\":{\"name\":\"Ivan\"}}'
        },
        'side': 2
      }
    ]
  }
}
```

Пример 120. Получение списка объектов, у которых нет никаких других указанных связей, кроме как с объектами со значением поля `name`, равным 'Ivan'

```
{
  'links': {
    'and': [
      {
        'domainId': 'example.com',
        'linkageId': 'ProjectContribution',
        'object': {
          'conditions': '{\"$every\":{\"name\":\"Ivan\"}}'
        },
        'side': 2
      }
    ]
  }
}
```

Создание объекта (`create`)

Метод используется для создания объекта. Набор атрибутов объекта и их значения задаются с помощью параметров.

Пример 121. Синтаксис TypeScript

```
create({
  domainId: <domain_id>, // string
  entityId: <entity_id>, // string
  values: <values>, // Map<string, any>
})
```

Пример 122. Синтаксис Python

```
create(
    domain_id=<domain_id>, # str
    entity_id=<entity_id>, # str
    values=<values>, # str
)
```

Параметры метода включают в себя:

- `<domain_id>` — идентификатор домена (строка);
- `<entity_id>` — идентификатор сущности (строка);
- `<values>` — набор пар "ключ — значение" с атрибутами объекта и их значениями (в TypeScript — коллекция Map, в Python — JSON-объект в виде строки).

Примеры создания объекта

Приведенные примеры для TypeScript отличаются способом передачи значений:

- В первом примере для создания Map используется метод `Object.entries()`.
- Во втором примере данные сохраняются в `mapForCreate` как массив массивов вида `['valueAttribute','value']`, где `valueAttribute` — это атрибут из схемы домена, `value` — значение атрибута.
- В третьем примере каждая пара атрибутов передается в отдельном вызове метода `mapForCreate.set()`.

▼ *Пример скрипта с `create` в TypeScript (`Object.entries()`)*

```
id: create_object_in_cms_ts_entries
name: 'create_object_in_cms_ts_entries'
version: 0.0.1
type: script
tags: [tests, typescript, bun, cms]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  const domainId = 'example.com'
  const entityId = 'Person'

  export async function main() {
    try {
      // Создание Map для передачи данных в функцию создания объекта.
      const mapForCreate = new Map<string, any>(
        Object.entries({
          department: 'IT department',
          mail: 'mail@example.com',
          ...
        }),
      )
```

```
      // Вызов функции создания объекта с передачей domainId, entityId и
mapForCreate.
      const cmsObjectToBeCreated = await sdk.cms.create({ domainId, entityId,
values: mapForCreate })

      // Вывод созданного объекта в консоль.
      console.log(cmsObjectToBeCreated)
    } catch (e) {
      // Вывод ошибки в консоль, если произошла ошибка при создании объекта.
      console.log(e)
    }
  }
```

▼ *Пример скрипта с `create` в TypeScript (массив массивов значений)*

```
id: create_object_in_cms_ts_arr
name: 'create_object_in_cms_ts_arr'
version: 0.0.1
type: script
tags: [tests, typescript, bun, cms]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  const domainId = 'example.com'
  const entityId = 'Person'

  export async function main() {
    try {
      // Создание Map для передачи данных в функцию создания объекта.
      const mapForCreate = new Map<string, any>([
        ['mail','mail@example.com'],
        ['department','IT department'],
        ...
      ])

      // Вызов функции создания объекта с передачей domainId, entityId и
mapForCreate.
      const cmsObjectToBeCreated = await sdk.cms.create({ domainId, entityId,
values: mapForCreate })

      // Вывод созданного объекта в консоль.
      console.log(cmsObjectToBeCreated)
    } catch (e) {
      // Вывод ошибки в консоль, если произошла ошибка при создании объекта.
      console.log(e)
    }
  }
```

▼ *Пример скрипта с `create` в TypeScript (`set`)*

```
id: create_object_in_cms_ts_set
name: 'create_object_in_cms_ts_set'
version: 0.0.1
type: script
tags: [tests, typescript, bun, cms]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  // Инициализация переменных domainId и entityId.
  const domainId = 'example.com'
  const entityId = 'Person'

  export async function main() {
    try {
      // Создание Map для передачи данных в функцию создания объекта.
      const mapForCreate = new Map<string, any>()
      mapForCreate.set('mail', 'mail@example.com')
      mapForCreate.set('department', 'IT department')
      ...

      // Вызов функции создания объекта с передачей domainId, entityId и
mapForCreate.
      const cmsObjectToBeCreated = await sdk.cms.create({ domainId, entityId,
values: mapForCreate })

      // Вывод созданного объекта в консоль.
      console.log(cmsObjectToBeCreated)
    } catch (e) {
      // Вывод ошибки в консоль, если произошла ошибка при создании объекта.
      console.log(e)
    }
  }
```

▼ *Пример скрипта с* `create` *в Python*

```
id: create_object_in_cms_py
name: create_object_in_cms_py
version: 0.0.1
type: script
tags: [tests, python, cms]
source: !py |
    from playbooks.cms import CmsClient, Filter, Sorting, Pagination, Order

    # Инициализация переменных domain_id и entity_id - id домена и сущности.
    domain_id = 'example.com'
    entity_id = 'Person'

    def main():
        try:
            client = CmsClient.initialize()
```

```
            id = client.create(domain_id=domain_id, entity_id=entity_id,
values='{'mail':'mail@example.com', 'department':'IT department', ...}').id  #
Создание объекта с указанными параметрами.

            print('Object created')  # Вывод сообщения об успешном создании объекта.

        except Exception as error:
            # Обработка и вывод ошибки, если произошла ошибка при создании объекта.
            print('An exception occurred:', error)
```

Примеры массового создания объектов

В приведенных примерах для массового создания экземпляров сущностей используется массив объектов с атрибутами и их значениями. Массив обрабатывается циклом, который поочередно вызывает метод `create` для каждого объекта.

▼ *Пример скрипта с массовым созданием объектов в TypeScript*

```
id: array_create_object_in_cms_ts
name: 'array_create_object_in_cms_ts'
version: 0.0.1
type: script
tags: [tests, typescript, bun, cms]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  const domainId = 'example.com'
  const entityId = 'Person'

  export async function main() {
    try {
      // Создание массива для передачи данных в функцию создания объекта.
      const objectsData = [
        { mail: 'mail@example.com', department: 'IT department', ... },
        { mail: 'mail2@example.com', department: 'Sales department', ... },
        ...,
      ]

      for (const item of objectsData) {
        const mapForCreate = new Map(Object.entries(item))

        // Вызов функции создания объекта с передачей domainId, entityId и
mapForCreate.
        const cmsObjectToBeCreated = await sdk.cms.create({ domainId, entityId,
values: mapForCreate })

        // Вывод созданного объекта в консоль.
        console.log(cmsObjectToBeCreated)
      }
    } catch (e) {
```

```
      // Вывод ошибки в консоль, если произошла ошибка при создании объекта.
      console.log(e)
    }
  }
```

▼ *Пример скрипта с массовым созданием объектов в Python*

```
id: array_create_object_in_cms_py
name: array_create_object_in_cms_py
version: 0.0.1
type: script
tags: [tests, python, cms]
source: !py |
    from playbooks.cms import CmsClient, Filter, Sorting, Pagination, Order
    import json

    # Инициализация переменных domain_id и entity_id - id домена и сущности.
    domain_id = 'example.com'
    entity_id = 'Person'

    def main():
        try:
            client = CmsClient.initialize()

            # Массив данных для создания объектов.
            objects_data = [
                { 'mail': 'mail@example.com', 'department': 'IT department', ... },
                { 'mail': 'mail2@example.com', 'department': 'Sales department', ...
},
                ...,
            ]

            for item in objects_data:
                json_data = json.dumps(item)

                # Создание объекта с указанными параметрами.
                object_id = client.create(domain_id=domain_id, entity_id=entity_id,
values=json_data).id

                print(f'Object created with ID: {object_id}')

        except Exception as error:
            # Обработка и вывод ошибки, если произошла ошибка при создании объекта.
            print('An exception occurred:', error)
```

Полное обновление объекта (`update`)

Метод используется для полного изменения набора атрибутов и значений объекта с сохранением его идентификатора.

Пример 123. Синтаксис TypeScript

```
update({
  domainId: <domain_id>, // string
  entityId: <entity_id>, // string
  id: <object_id>, // string
  values: <values>, // Map<string, any>
})
```

Пример 124. Синтаксис Python

```
update(
    domain_id=<domain_id>, # str
    entity_id=<entity_id>, # str
    id=<object_id>, # str
    values=<values>, # str
)
```

Параметры метода включают в себя:

- `<domain_id>` — идентификатор домена (строка);
- `<entity_id>` — идентификатор сущности (строка);
- `<object_id>` — идентификатор объекта (строка);
- `<values>` — набор пар "ключ — значение" с атрибутами объекта и их значениями (в TypeScript — коллекция Map, в Python — JSON-объект в виде строки).

Примеры обновления объекта

Приведенные примеры для TypeScript отличаются способом передачи значений:

- В первом примере для создания Map используется метод `Object.entries()`.
- Во втором примере данные сохраняются в `mapForUpdate` как массив массивов вида `['valueAttribute','value']`, где `valueAttribute` — это атрибут из схемы домена, `value` — значение атрибута.
- В третьем примере каждая пара атрибутов передается в отдельном вызове метода `map.set()`.

▼ *Пример скрипта с `update` в TypeScript (`Object.entries()`)*

```
id: update_object_in_cms_ts_arr
name: 'update_object_in_cms_ts_arr'
version: 0.0.1
type: script
tags: [tests, typescript, bun, cms]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  // Инициализация переменных domainId, entityId и objectId - id экземпляра
сущности.
```

```
  const domainId = 'example.com'
  const entityId = 'Person'
  const objectId = 'a1b2c3d4-5678-90ab-cdef-1234567890ab'

  export async function main() {
    try {
      // Создание Map для передачи данных в функцию обновления объекта.
      const mapForUpdate = new Map<string, any>(
        Object.entries({
          mail: 'mail@example.com',
          department: 'IT department',
          ...
        })
      )

      // Вызов функции обновления объекта с передачей objectId, domainId, entityId и
mapForUpdate.
      const cmsObjectToBeUpdated = await sdk.cms.update({ domainId, entityId, id:
objectId, values: mapForUpdate })

      // Вывод обновленного объекта в консоль.
      console.log(cmsObjectToBeUpdated)
    } catch (error) {
      // Вывод ошибки в консоль, если произошла ошибка при обновлении объекта.
      console.log(error)
    }
  }
```

▼ *Пример скрипта с* `update` *в TypeScript (массив массивов значений)*

```
id: update_object_in_cms_ts_arr
name: 'update_object_in_cms_ts_arr'
version: 0.0.1
type: script
tags: [tests, typescript, bun, cms]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  // Инициализация переменных domainId, entityId и objectId - id экземпляра
сущности.
  const domainId = 'example.com'
  const entityId = 'Person'
  const objectId = 'a1b2c3d4-5678-90ab-cdef-1234567890ab'

  export async function main() {
    try {
      // Создание Map для передачи данных в функцию обновления объекта.
      const mapForUpdate = new Map<string, any>([
        ['mail', 'new_mail@example.com'],
        ['department', 'IT department'],
```

```
        ...
      ])

      // Вызов функции обновления объекта с передачей objectId, domainId, entityId и
mapForUpdate.
      const cmsObjectToBeUpdated = await sdk.cms.update({ domainId, entityId, id:
objectId, values: mapForUpdate })

      // Вывод обновленного объекта в консоль.
      console.log(cmsObjectToBeUpdated)
    } catch (error) {
      // Вывод ошибки в консоль, если произошла ошибка при обновлении объекта.
      console.log(error)
    }
  }
```

▼ *Пример скрипта с* `update` *в TypeScript (*`set`*)*

```
id: update_object_in_cms_ts_set
name: 'update_object_in_cms_ts_set'
version: 0.0.1
type: script
tags: [tests, typescript, bun, cms]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  // Инициализация переменных domainId, entityId и objectId - id экземпляра
сущности.
  const domainId = 'example.com'
  const entityId = 'Person'
  const objectId = 'a1b2c3d4-5678-90ab-cdef-1234567890ab'

  export async function main() {
    try {
      // Создание Map для передачи данных в функцию обновления объекта.
      const map = new Map<string, any>()
      map.set('mail', 'new_mail@example.com')
      map.set('department', 'IT department')
      ...

      // Вызов функции обновления объекта с передачей objectId, domainId, entityId и
map.
      const cmsObjectToBeUpdated = await sdk.cms.update({ domainId, entityId, id:
objectId, values: map })

      // Вывод обновленного объекта в консоль.
      console.log(cmsObjectToBeUpdated)
    } catch (error) {
      // Вывод ошибки в консоль, если произошла ошибка при обновлении объекта.
      console.log(error)
```

```
    }
  }
```

▼ *Пример скрипта с* `update` *в Python*

```
id: update_object_in_cms_py
name: update_object_in_cms_py
version: 0.0.1
type: script
tags: [tests, python, cms]
source: !py |
    from playbooks.cms import CmsClient, Filter, Sorting, Pagination, Order

    # Инициализация переменных domain_id, entity_id и object_id - id домена,
сущности и объекта.
    domain_id = 'example.com'
    entity_id = 'Person'
    object_id = 'a1b2c3d4-5678-90ab-cdef-1234567890ab'

    def main():
        try:
            client = CmsClient.initialize()
            client.update(id=object_id, domain_id=domain_id, entity_id=entity_id,
values='{'mail':'new_mail@example.com', 'department':'IT department', ... }') #
Обновление объекта с указанными параметрами и новым значением атрибута.

            cms_object = client.get(id=object_id, domain_id=domain_id,
entity_id=entity_id)  # Получение обновленного объекта.

            print('Print updated object:') # Вывод сообщения об успешном обновлении
объекта и его значений.
            print(cms_object)

        except Exception as error:
            # Обработка и вывод ошибки, если произошла ошибка при обновлении
объекта.
            print('An exception occurred:', error)
```

Примеры массового обновления объектов

В приведенных примерах для массового обновления экземпляров сущностей используется массив объектов с атрибутами и их значениями. Массив обрабатывается циклом, который поочередно вызывает метод `update` для каждого объекта.

▼ *Пример скрипта с массовым обновлением объектов в TypeScript*

```
id: array_update_object_in_cms_ts
name: 'array_update_object_in_cms_ts'
version: 0.0.1
type: script
```

```
tags: [tests, typescript, bun, cms]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  const domainId = 'example.com'
  const entityId = 'Person'

  export async function main() {
    try {
      // Создание массива для передачи данных в функцию обновления объекта.
      const objectsData = [
        { id: 'a1b2c3d4-5678-90ab-cdef-1234567890ab', values: { mail:
'new_mail@example.com', department: 'IT department', ..., } },
        { id: 'f1e2d3c4-b567-89a0-bcde-0987654321ff', values: { mail:
'new_mail2@example.com', department: 'Sales department', ..., } },
        ...,
      ]

      for (const item of objectsData) {
        const mapForUpdate = new Map(Object.entries(item.values))

        // Вызов функции обновления объекта с передачей domainId, entityId и
mapForUpdate.
        const cmsObjectToBeUpdated = await sdk.cms.update({ domainId, entityId, id:
item.id, values: mapForUpdate })

        // Вывод обновленного объекта в консоль.
        console.log(cmsObjectToBeUpdated)
      }
    } catch (e) {
      // Вывод ошибки в консоль, если произошла ошибка при обновлении объекта.
      console.log(e)
    }
  }
```

▼ *Пример скрипта с массовым обновлением объектов в Python*

```
id: array_update_object_in_cms_py
name: array_update_object_in_cms_py
version: 0.0.1
type: script
tags: [tests, python, cms]
source: !py |
    from playbooks.cms import CmsClient, Filter, Sorting, Pagination, Order
    import json

    # Инициализация переменных domain_id и entity_id - id домена и сущности.
    domain_id = 'example.com'
    entity_id = 'Person'
```

```
    def main():
        try:
            client = CmsClient.initialize()

            # Массив данных для обновления объектов.
            objects_data = [
                { id: 'a1b2c3d4-5678-90ab-cdef-1234567890ab', values:
'{'mail':'new_mail@example.com','department':'IT department', ... }' },
                { id: 'f1e2d3c4-b567-89a0-bcde-0987654321ff', values:
'{'mail':'new_mail2@example.com','department':'Sales department', ... }' },
                 ...,
            ]

            for item in objects_data:
                object_id = item.get('id')
                entity_values = item.get('values')

                # Обновление объекта с указанными параметрами.
                client.update(id=object_id, domain_id=domain_id,
entity_id=entity_id, values=entity_values)

                cms_object = client.get(id=object_id, domain_id=domain_id,
entity_id=entity_id)  # Получение обновленного объекта.

                print('Print updated object:') # Вывод сообщения об успешном
обновлении объекта и его значений.
                print(cms_object)

        except Exception as error:
            # Обработка и вывод ошибки, если произошла ошибка при обновлении
объекта.
            print('An exception occurred:', error)
```

Частичное изменение объекта (`patch`)

Метод используется для изменения отдельных атрибутов и значений объекта с сохранением его идентификатора.

Пример 125. Синтаксис TypeScript

```
patch({
  attributeIds: <attribute_ids>, // string[]
  delta: <delta>, // string
  domainId: <domain_id>, // string
  entityId: <entity_id>, // string
  id: <object_id>, // string
})
```

Пример 126. Синтаксис Python

```
patch(
    domain_id=<domain_id>, # str
    entity_id=<entity_id>, # str
    id=<object_id>, # str
    attribute_ids=<attribute_ids>, # List[str]
    delta=<delta>, # str
)
```

Параметры метода включают в себя:

- `<domain_id>` — идентификатор домена (строка);
- `<entity_id>` — идентификатор сущности (строка);
- `<object_id>` — идентификатор объекта (строка);
- `<attribute_ids>` — идентификаторы атрибутов объекта, которые нужно изменить (массив строк);
- `<delta>` — набор пар "ключ — значение" с атрибутами объекта и их новыми значениями (JSON-объект в виде строки).

Примеры изменения объекта

▼ *Пример скрипта с `patch` в TypeScript*

```
id: patch_object_in_cms_ts
name: 'patch_object_in_cms_ts'
version: 0.0.1
type: script
tags: [tests, typescript, bun, cms]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  // Инициализация переменных domainId, entityId и objectId - id экземпляра
сущности.
  const domainId = 'example.com'
  const entityId = 'Person'
  const objectId = 'a1b2c3d4-5678-90ab-cdef-1234567890ab'

  export async function main() {
    try {
      // Вызов функции изменения объекта с передачей objectId, domainId и entityId.
      const cmsObjectToBePatched = await sdk.cms.patch({
        attributeIds: ['mail'],
        delta: '{"mail":"new_mail@example.com"}',
        domainId,
        entityId,
        id: objectId,
      })

      // Вывод объекта в консоль.
```

```
      console.log(cmsObjectToBePatched)
    } catch (error) {
      // Вывод ошибки в консоль, если произошла ошибка при изменении объекта.
      console.log(error)
    }
  }
```

▼ *Пример скрипта с* `patch` *в Python*

```
id: patch_object_in_cms_py
name: patch_object_in_cms_py
version: 0.0.1
type: script
tags: [tests, python, cms]
source: !py |
    from playbooks.cms import CmsClient, Filter, Sorting, Pagination, Order

    domain_id = 'example.com'
    entity_id = 'Person'
    object_id= 'a1b2c3d4-5678-90ab-cdef-1234567890ab'

    def main():
        try:
          client = CmsClient.initialize()

          # Частичное изменение объекта с указанными параметрами и новым значением
атрибута.
          client.patch(id=object_id, domain_id=domain_id, entity_id=entity_id,
attribute_ids=['mail'], delta='{"mail":"new_mail@example.com"}')

          cms_object = client.get(id=object_id, domain_id=domain_id,
entity_id=entity_id) # Получение обновленного объекта.

          print('Print patched object:') # Вывод сообщения об успешном применении
частичного изменения к объекту и его значений.
          print(cms_object)

        except Exception as error:
            # Обработка и вывод ошибки, если произошла ошибка при изменении объекта.
            print('An exception occurred:', error)
```

Примеры массового редактирования объектов

В приведенных примерах для массового редактирования объектов используется массив, содержащий атрибуты и их значения. Затем этот массив обрабатывается циклом, который поочередно вызывает метод `patch` для каждого объекта в массиве.

▼ *Пример скрипта с массовым редактированием объектов в TypeScript*

```
id: array_patch_object_in_cms_ts
```

```
name: 'array_patch_object_in_cms_ts'
version: 0.0.1
type: script
tags: [tests, typescript, bun, cms]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  const domainId = 'example.com'
  const entityId = 'Person'

  export async function main() {
    try {
      // Создание массива для передачи данных в функцию редактирования объекта.
      const objectsData = [
        { id: 'a1b2c3d4-5678-90ab-cdef-1234567890ab', attributeIds: ['mail'],
values: '{'mail':'new_mail@example.com'}' },
        { id: 'f1e2d3c4-b567-89a0-bcde-0987654321ff', attributeIds: ['mail'],
values: '{'mail':'new_mail2@example.com'}' },
        ...,
      ]

      for (const item of objectsData) {
        // Вызов функции обновления объекта с передачей domainId, entityId и id
объекта.
        const cmsObjectToBePatched = await sdk.cms.patch({
          attributeIds: item.attributeIds,
          delta: item.values,
          domainId,
          entityId,
          id: item.id,
        })

        // Вывод объекта в консоль.
        console.log(cmsObjectToBeUpdated)
      }
    } catch (e) {
      // Вывод ошибки в консоль, если произошла ошибка при изменении объекта.
      console.log(e)
    }
  }
```

▼ *Пример скрипта с массовым редактированием объектов в Python*

```
id: array_patch_object_in_cms_py
name: array_patch_object_in_cms_py
version: 0.0.1
type: script
tags: [tests, python, cms]
source: !py |
    from playbooks.cms import CmsClient, Filter, Sorting, Pagination, Order
```

```
    import json

    # Инициализация переменных domain_id и entity_id - id домена и сущности.
    domain_id = 'example.com'
    entity_id = 'Person'

    def main():
        try:
            client = CmsClient.initialize()

            # Массив данных для редактирования объектов.
            objects_data = [
                { id: 'a1b2c3d4-5678-90ab-cdef-1234567890ab', attribute_ids:
['mail'], values: '{'mail':'new_mail@example.com'}' },
                { id: 'f1e2d3c4-b567-89a0-bcde-0987654321ff', attribute_ids:
['mail'], values: '{'mail':'new_mail2@example.com'}' },
                 ...,
            ]

            for item in objects_data:
                object_id = item.get('id')
                attribute_ids = item.get('attribute_ids')
                entity_values = item.get('values')

                # Частичное изменение объекта с указанными параметрами и новым
значением атрибута.
                client.patch(id=object_id, domain_id=domain_id, entity_id=entity_id,
attribute_ids=attribute_ids, delta=entity_values)

                cms_object = client.get(id=object_id, domain_id=domain_id,
entity_id=entity_id) # Получение обновленного объекта.

                print('Print patched object:') # Вывод сообщения об успешном
применении частичного изменения к объекту и его значений.
                print(cms_object)

        except Exception as error:
            # Обработка и вывод ошибки, если произошла ошибка при изменении объекта.
            print('An exception occurred:', error)
```

Удаление объекта (`delete`)

Метод используется для удаления объекта.

Пример 127. Синтаксис TypeScript

```
delete({
  domainId: <domain_id>, // string
  entityId: <entity_id>, // string
  id: <object_id> // string
```

```
})
```

Пример 128. Синтаксис Python

```
delete(
    domain_id=<domain_id>, # str
    entity_id=<entity_id>, # str
    id=<object_id> # str
)
```

Параметры метода включают в себя:

- `<domain_id>` — идентификатор домена (строка);
- `<entity_id>` — идентификатор сущности (строка);
- `<object_id>` — идентификатор объекта (строка).

Примеры удаления объекта

▼ *Пример скрипта с `delete` в TypeScript*

```
id: delete_object_in_cms_ts
name: 'delete_object_in_cms_ts'
version: 0.0.1
type: script
tags: [tests, typescript, bun, cms]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  // Инициализация переменных domainId, entityId и objectId - id экземпляра
сущности.
  const domainId = 'example.com'
  const entityId = 'Person'
  const objectId = 'a1b2c3d4-5678-90ab-cdef-1234567890ab'

  export async function main() {
    try {
      // Вызов функции удаления объекта с передачей objectId, domainId и entityId.
      const cmsObjectToBeDeleted = await sdk.cms.delete({ domainId, entityId, id:
objectId })

      // Вывод объекта в консоль.
      console.log('Result is Ok. Deleted object:')
      console.log(cmsObjectToBeDeleted)
    } catch (error) {
      // Вывод ошибки в консоль, если произошла ошибка при удалении объекта.
      console.log(error)
    }
  }
```

▼ *Пример скрипта с `delete` в Python*

```
id: delete_object_in_cms_py
name: delete_object_in_cms_py
version: 0.0.1
type: script
tags: [tests, python, cms]
source: !py |
    from playbooks.cms import CmsClient, Filter, Sorting, Pagination, Order

    # Инициализация переменных domain_id, entity_id и object_id - id домена,
сущности и объекта.
    domain_id = 'example.com'
    entity_id = 'Person'
    objectId = 'a1b2c3d4-5678-90ab-cdef-1234567890ab'

    def main():
        try:
            client = CmsClient.initialize()
            deleted_object = client.delete(id=objectId, domain_id=domain_id,
entity_id=entity_id) # Удаление объекта с указанными параметрами.

            print(f'Deletion successful for object: {deleted_object}') # Вывод
сообщения об успешном удалении объекта.

        except Exception as error:
            # Обработка и вывод ошибки, если произошла ошибка при удалении объекта.
            print('An exception occurred:', error)
```

Примеры массового удаления объектов

В приведенных примерах для массового удаления объектов используется массив идентификаторов объектов. Этот массив обрабатывается циклом, который поочередно вызывает метод `delete` для каждого объекта.

▼ *Пример скрипта с массовым удалением объектов в TypeScript*

```
id: array_delete_object_in_cms_ts
name: 'array_delete_object_in_cms_ts'
version: 0.0.1
type: script
tags: [tests, typescript, bun, cms]
source: !bun |
  import * as sdk from '@playbooks/sdk'

  // Инициализация переменных domainId и entityId.
  const domainId = 'example.com'
  const entityId = 'Person'

  export async function main() {
    try {
```

```
      const objectsList = [
        'a1b2c3d4-5678-90ab-cdef-1234567890ab',
        'f1e2d3c4-b567-89a0-bcde-0987654321ff',
        ...,
      ]

      for (const item of objectsList) {
        // Вызов функции удаления объекта с передачей objectId, domainId и id
объекта.
        const cmsObjectToBeDeleted = await sdk.cms.delete({ domainId, entityId, id:
item })

        // Вывод объекта в консоль.
        console.log('Result is Ok. Deleted object:')
        console.log(cmsObjectToBeDeleted)
       }
    } catch (error) {
      // Вывод ошибки в консоль, если произошла ошибка при удалении объекта.
      console.log(error)
    }
  }
```

▼ *Пример скрипта с массовым удалением объектов в Python*

```
id: array_delete_object_in_cms_py
name: array_delete_object_in_cms_py
version: 0.0.1
type: script
tags: [tests, python, cms]
source: !py |
    from playbooks.cms import CmsClient, Filter, Sorting, Pagination, Order

    # Инициализация переменных domain_id, entity_id - id домена и сущности.
    domain_id = 'example.com'
    entity_id = 'Person'

    def main():
        try:
            client = CmsClient.initialize()

            const objects_list = [
                'a1b2c3d4-5678-90ab-cdef-1234567890ab',
                'f1e2d3c4-b567-89a0-bcde-0987654321ff',
                ...
            ]

            for item in objects_list:
                deleted_object = client.delete(id=item, domain_id=domain_id,
entity_id=entity_id) # Удаление объекта с указанными параметрами.
```

```
                print(f'Deletion successful for object: {deleted_object}') # Вывод
сообщения об успешном удалении объекта.

        except Exception as error:
            # Обработка и вывод ошибки, если произошла ошибка при удалении объекта.
            print('An exception occurred:', error)
```

Отображение ошибок при работе с пакетом SDK

Ранее при запуске любого метода система возвращала информацию об объекте в виде полей, упакованных в структуру Rust. Теперь система возвращает непосредственно информацию о данных объекта. Возможную ошибку, возвращаемую методом, можно перехватить с помощью стандартной конструкции TypeScript `try···catch`.

Использование API для активных списков

Если планируется добавление значений объектов в активные списки, необходимо выполнить соответствующие шаги настройки API.

Клиент `activeList`, содержащийся в пакете `@playbooks/sdk`, позволяет работать с активными списками с использованием описанных ниже методов.

Методы API для работы с активными списками включают:

- Вывод отдельной записи активного списка по ключу, а также массива записей активного списка.
- Добавление записи активного списка.
- Удаление записи активного списка.

Описания и примеры приведены для TypeScript (среда выполнения Bun) и Python.

Подготовка клиента `activeList` для работы с активными списками

Чтобы начать работу с активными списками, используя пакет SDK, импортируйте его:

Пример 129. Синтаксис импорта в TypeScript

```
import { SetupActiveList } from '@playbooks/active-list-client'
...
const activeClient = SetupActiveList()
...
export async function main() {
  const v = await activeClient.<method_name>(<arg_1>, <arg_2>, ..., <arg_m>)
  console.log(v)
}
```

Пример 130. Синтаксис импорта в Python

```
from playbooks.active_list import ActiveListClient

...
```

```
def main() {
...
    client = ActiveListClient.initialize()
    v = client.<method_name>(<arg_1>, <arg_2>, ..., <arg_m>)
    print(v)
...
}
```

Здесь:

- `<method_name>` — имя используемого метода;
- `<arg_N>` — параметры метода.

Вывод записи активного списка по ключу (`get_record`)

Пример 131. Синтаксис TypeScript

```
get_record({
  activeListId: <active_list_id>, // string
  key: <key> // string
})
```

Пример 132. Синтаксис Python

```
get_record(
    id=<active_list_id>, # str
    key=<key> # str
)
```

Параметры метода включают в себя:

- `<active_list_id>` — идентификатор активного списка (строка);
- `<key>` — ключ записи (строка).

Примеры получения записи активного списка

▼ *Пример скрипта с `get_record` в TypeScript*

```
id: get_active_list_object_bun
name: Get active list object bun
description: Получение записи активного списка скриптом Bun (TypeScript) с помощью
метода get_record
author: John Doe
tags: [bun, typescript, active list]
version: 0.0.1
type: script
source: !bun |
  import { SetupActiveList } from '@playbooks/active-list-client'
```

```
  const activeListId = '95aaab6a-9119-485e-8621-c5ff4dbc1a40' // id активного
списка.

  export async function main() {
    try {
      // Настройка клиента активного списка.
      const activeClient = SetupActiveList()

      // Вывод id активного списка.
      console.log(`Получение значений из активного списка с id: ${activeListId}`)

      // Получение записи активного списка.
      const activeListObject = await activeClient.get_record({activeListId, key:
'1'})

      console.log('Активный список получен успешно:')

      // Вывод значений из ответа.
      console.log('count:', activeListObject.count)
      console.log('created_at:', activeListObject.created_at)
      console.log('expired_at:', activeListObject.expired_at)
      console.log('updated_at:', activeListObject.updated_at)
      console.log('fields:', activeListObject.fields)
      console.log('key:', activeListObject.key)

      // Или вывод полного ответа из запроса:
      // console.log('Полный ответ:', JSON.stringify(activeListObject, null, 2))
    } catch (error) {
      console.error(`Произошла ошибка: ${error.message}`)
    }
  }
```

▼ *Пример скрипта с* `get_record` *в Python*

```
id: get_active_list_object_py
name: Get active list object py
description: Получение записи активного списка скриптом Python с помощью метода
get_record
author: John Doe
tags: [python, active list]
version: 0.0.1
type: script
source: !py |
    from playbooks.active_list import ActiveListClient

    active_list_id = '95aaab6a-9119-485e-8621-c5ff4dbc1a40' # id активного списка.

    def main():
        try:
            client = ActiveListClient.initialize()
```

```
            record = client.get_record(id=active_list_id, key='1') # id активного
списка и значение ключа.
            print('Получение записи')
            print('-' * 30)
            print('Запись с ключом', record.key)
            print('Значение', record.fields)

        except Exception as error:
            # Обработка исключения.
            logger.error(f'Произошла ошибка: {error}')
```

Вывод активного списка (`find_records`)

Пример 133. Синтаксис TypeScript

```
find_records({
  activeListId: <active_list_id> // string
})
```

Пример 134. Синтаксис Python

```
find_records(
    id=<active_list_id> # str
)
```

Параметры метода включают в себя:

- `<active_list_id>` — идентификатор активного списка.

Примеры получения записи активного списка

▼ *Пример скрипта с `find_records` в TypeScript*

```
id: Find_active_list_object_bun
name: Find active list object bun
description: Получение массива записей активного списка скриптом Bun (TypeScript) с
помощью метода find_records
author: John Doe
tags: [typescript, bun, active list]
version: 0.0.1
type: script
source: !bun |
  import { SetupActiveList } from '@playbooks/active-list-client'

  const activeListId = '5aaab6a-9119-485e-8621-c5ff4dbc1a40' // id созданного
активного списка.

  export async function main() {
    try {
```

```
      // Настройка клиента активного списка.
      const activeClient = SetupActiveList()

      console.log(`Получение списка объектов со значениями из активного списка с id:
${activeListId}`)

      // Поиск записей активного списка. Возвращает Result<Array<ActiveListRecord>>.
      const records = await activeClient.find_records({activeListId})

      console.log('Список записей в активном списке')
      console.log('-'.repeat(30))

      for (const record of records) {
        console.log('Запись с ключом:', record.key)
        console.log('Значение:', record.fields)
        console.log('-'.repeat(30)) // Разделитель для удобства чтения.

      } catch (error) {
        console.error(`Произошла ошибка: ${error.message}`)
      }
  }
```

▼ *Пример скрипта с* `find_records` *в Python*

```
id: list_active_list_object_py
name: List active list object py
description: Получение массива записей активного списка скриптом Python с помощью
метода find_records
author: John Doe
tags: [python, active list]
version: 0.0.1
type: script
source: !py |
    from playbooks.active_list import ActiveListClient

    active_list_id = '6c707e96-f786-4d84-b787-3956d1b4e7a1' # id созданного
активного списка.

        def main():
            try:
                client = ActiveListClient.initialize()

                print(f'Получение списка объектов со значениями из активного списка
с id: {active_list_id}')

                records = client.find_records(id=active_list_id)
                print('Список записей в активном списке')
                print('-' * 30)
                for record in records:
                    print('Запись с ключом', record.key)
```

```
                    print('Значение', record.fields)
                    print('-' * 30)  # Разделитель для удобства чтения.

            except Exception as error:
                # Обработка исключения.
                logger.error(f'Произошла ошибка: {error}')
```

Создание записи в активном списке (`insert-record`)

Метод используется для создания записи активного списка. Значения записи активного списка задаются с помощью параметров.

Пример 135. Синтаксис TypeScript

```
insert_record({
  activeListId: <active_list_id>, // string
  key: <key>, // Entry[]
  value: <values>, // Entry[]
})
```

Пример 136. Синтаксис Python

```
insert_record(
    id=<active_list_id>, # str
    key=<key>, # Dict[str, str | int | float | bool | Timestamp]
    value=<values>, # Dict[str, str | int | float | bool | Timestamp]
)
```

Параметры метода включают в себя:

- `<active_list_id>` — идентификатор активного списка (строка);
- `<key>` — ключ записи (в TypeScript — массив пар "ключ — значение", в Python — словарь);
- `<values>` — поля записи и их значения (в TypeScript — массив пар "ключ — значение", в Python — словарь).

Пример 137. Синтаксис пары "ключ — значение"

```
{
  key: <field_name>, // string
  value: {
    <value_type>: <field_value> // number | boolean | string | { nanos: number,
seconds: number} | null
  }
}
```

Здесь:

- `<field_name>` — имя поля (строка);

- `<value_type>` — тип значения (`Integer`, `Float`, `Boolean`, `String`, `Timestamp`, `Empty`);
- `<field_value>` — значение поля.

Примеры создания записи активного списка

▼ *Пример скрипта с `insert_record` в TypeScript*

```
id: create_active_list_object_bun
name: Create active list object bun
description: Создание записи активного списка скриптом Bun (TypeScript) с помощью
метода insert_record
author: John Doe
tags: [bun, typescript, active list]
version: 0.0.1
type: script
source: !bun |
  import { SetupActiveList } from '@playbooks/active-list-client'
  const activeListId = '95aaab6a-9119-485e-8621-c5ff4dbc1a40' // id активного
списка.
  export async function main() {
    try {
      // Настройка клиента активного списка.
      const activeClient = SetupActiveList()

      // Вывод id активного списка.
      console.log(`Создание в активном списке с id: ${activeListId}`)

      // Создание записей активного списка. Возвращает массив строк с результатами
создания. Принимает id, ключ, значение.
      const insertedObject = await activeClient.insert_record({
        activeListId,
        key: [
          {
            key: 'account', // Это ключ.
            value: {
              String: '1'
            }
          }
        ],
        value: [
          {
            key: 'commonName',
            value: {
              String: 'User'
            }
          },
          {
            key: 'upn',
            value: {
              String: 'john.doe@corp.local'
            }
```

```
          },
          {
            key: 'cmsObjectId',
            value: {
              String: '5'
            }
          }
        ]
      })

      console.log(`Создан объект с ключом: ${insertedObject}`)
    } catch (err) {
      console.error(`Произошла ошибка: ${err.message}`)
    }
  }
```

▼ *Пример скрипта с* `insert_record` *в Python*

```
id: create_active_list_object_py
name: Create active list object py
description: Создание записи активного списка скриптом Python с помошью метода
insert_record
author: John Doe
tags: [python, active list]
version: 0.0.1
type: script
source: !py |
    from playbooks.active_list import ActiveListClient

    active_list_id = '95aaab6a-9119-485e-8621-c5ff4dbc1a40' # id активного списка.

    def main():
        try:
            client = ActiveListClient.initialize()

            # Ключ - значение в активном списке.
            key = {
                'account': '1'
            }
            value = {
                'upn': 'john.doe@corp.local',
                'commonName': 'User',
                'cmsObjectId': '5'
            }
            client.insert_record(id=active_list_id, key=key, value=value)
            print(f'Запись создана в {active_list_id}')

        except Exception as error:
            # Обработка исключения.
            logger.error(f'Произошла ошибка: {error}')
```

Удаление записи активного списка (`delete_record`)

Пример 138. Синтаксис TypeScript

```
delete_record({
  activeListId: <active_list_id>, // string
  key: <key> // string
})
```

Пример 139. Синтаксис Python

```
delete_record(
    id=<active_list_id>, # str
    key=<key> # str
)
```

Параметры метода включают в себя:

- `<active_list_id>` — идентификатор активного списка (строка);
- `<key>` — ключ записи (строка).

Примеры удаления записи активного списка

▼ *Пример скрипта с `delete_record` в TypeScript*

```
id: delete_active_list_object_bun
name: Delete active list object bun
description: Удаление записей активного списка скриптом Bun (TypeScript) с помощью
метода delete_record
author: John Doe
tags: [bun, typescript, active list]
version: 0.0.1
type: script
source: !bun |
  import { SetupActiveList } from '@playbooks/active-list-client'

  const activeListId = '95aaab6a-9119-485e-8621-c5ff4dbc1a40'
  const activeListKeys = ['first_key', 'second_key', 'third_key', ...]

  export async function main() {
      try {
        // Настройка клиента активного списка.
        const activeClient = SetupActiveList()

        // Вывод id активного списка.
        console.log(`Удаление объектов из активного списка с id: ${activeListId}`)

        // Удаление записей из активного списка по каждому ключу.
        for (const key of activeListKeys) {
          // Удаление одной записи из активного списка.
```

```
          const deletedObject = await activeClient.delete_record({activeListId,
key})

          console.log(`Объект с ключом '${key}' удален.`)
        }
      } catch (error) {
        console.error(`Произошла ошибка: ${error.message}`)
      }
    }
```

▼ *Пример скрипта с* `delete_record` *в Python*

```
id: delete_active_list_object_py
name: Delete active list object py
description: Удаление записи активного списка скриптом Python с помощью метода
delete_record
author: John Doe
tags: [python, active list]
version: 0.0.1
type: script
source: !py |
    from playbooks.active_list import ActiveListClient

    active_list_id = '95aaab6a-9119-485e-8621-c5ff4dbc1a40' # id активного списка.
    key_id = 'account' # Ключ, по которому производится поиск записи.

    def main():
        try:
            client = ActiveListClient.initialize()

            # Удаление записи.
            client.delete_record(id=active_list_id, key=key_id)
            print(f'Запись с ключом {key_id} удалена')

        except Exception as error:
            # Обработка исключения.
            logger.error(f'Произошла ошибка: {error}')
```

Использование API для схем расчетов

Клиент `calculation`, содержащийся в пакете `@playbooks/sdk`, позволяет работать со схемами расчетов с использованием описанных ниже методов.

Для схем расчетов доступен только SDK на TypeScript.

Методы API для работы со схемами расчетов включают:

- Вывод результата выполненного расчета.
- Выполнение расчета.

- Выполнение обобщающего расчета.

Подготовка клиента `calculation` для работы со схемами расчетов

Чтобы начать работу со схемами расчетов, используя пакет SDK, импортируйте его:

Пример 140. Синтаксис импорта в TypeScript

```
import * as sdk from "@playbooks/sdk"
...
export async function main() {
...
  const v = await sdk.calculation.<method_name>(<args>)
  console.log(v)
...
}
```

Здесь:

- `<method_name>` — имя используемого метода;
- `<args>` — параметры метода.

Вывод результата выполненного расчета (`calculation`)

Пример 141. Синтаксис TypeScript

```
calculation({
  calculationId: <calculation_id>, // string
  calculationSchemaId: <calculation_schema_id>, // string
  objectId: <object_id> // string
})
```

Параметры метода включают в себя:

- `<calculation_id>` — идентификатор расчета;
- `<calculation_schema_id>` — идентификатор схемы расчетов;
- `<object_id>` — идентификатор объекта, для которого выполнен расчет.

Примеры получения результатов расчетов

▼ *Пример скрипта с `calculation` в TypeScript*

```
id: get_calculation_result_bun
name: Get calculation result bun
description: Получение результатов расчета скриптом Bun (TypeScript) с помощью
метода calculation
author: John Doe
tags: [bun, typescript, calculation]
version: 0.0.1
type: script
```

```
source: !bun |
  import * as sdk from '@playbooks/sdk';
  export async function main() {
    const command = {
      calculationId: 'test',
      calculationSchemaId: 'calculate_test',
      objectId: '551471a6-8bd2-4af7-a7ad-7af5432c9bea',
    }

    const { result } = await sdk.calculation.calculation(command);
    console.log('result', result);
    return result;
  }
```

Выполнение расчета (exec)

Пример 142. Синтаксис TypeScript

```
exec({
  calculationId: <calculation_id>, // string
  calculationSchemaId: <calculation_schema_id>, // string
  object: {
    domainId: <domain_id>, // string
    entityId: <entity_id>, // string
    id: <id> // string
  },
  cacheFirst: <cache_first> // boolean
})
```

Параметры метода включают в себя:

- `<calculation_id>` — идентификатор расчета;
- `<calculation_schema_id>` — идентификатор схемы расчетов;
- `<domain_id>` — идентификатор домена, для объекта которого нужно выполнить расчет;
- `<entity_id>` — идентификатор сущности в домене, для объекта которой нужно выполнить расчет;
- `<id>` — идентификатор объекта, для которого нужно выполнить расчет;
- `<cache_first>` — логическая переменная: если установить значение `true`, то расчет не будет выполнен заново, если его результат будет найден в кэше; если установить значение `false`, то расчет будет выполнен заново в любом случае.

Примеры выполнения расчета

▼ *Пример скрипта с `exec` в TypeScript*

```
id: exec_calculation_bun
name: Exec calculation bun
description: Выполнение расчета скриптом Bun (TypeScript) с помощью метода exec
author: John Doe
```

```
tags: [bun, typescript, calculation]
version: 0.0.1
type: script
source: !bun |
  import * as sdk from '@playbooks/sdk';
  export async function main() {
    const command = {
      calculationSchemaId: "calculate_test",
      calculationId: "test",
      object: {
        domainId: "evo.entity.category.standards",
        entityId: "Requirement",
        id: "551471a6-8bd2-4af7-a7ad-7af5432c9bea"
      },
      cacheFirst: false
    }

    const result = await sdk.calculation.exec(command);
    console.log('result', result);
    return result;
    }
```

Выполнение обобщающего расчета (`execSummary`)

Пример 143. Синтаксис TypeScript

```
execSummary({
  summaryCalculationId: <summary_calculation_id>, // string
  summaryCalculationSchemaId: <summary_calculation_schema_id>, // string
  calculations: [
    {
      calculationId: <calculation_id>, // string
      calculationSchemaId: <calculation_schema_id>, // string
      object: {
        domainId: <domain_id>, // string
        entityId: <entity_id>, // string
        id: <id> // string
      },
      cacheFirst: <cache_first> // boolean
    },
  ...
  ]
})
```

Параметры метода включают в себя:

- `<summary_calculation_id>` — идентификатор обобщающего расчета;
- `<summary_calculation_schema_id>` — идентификатор схемы, в которой определен обобщающий расчет;

- `calculations` — массив расчетов, входящих в обобщающий расчет:
  - `<calculation_id>` — идентификатор расчета;
  - `<calculation_schema_id>` — идентификатор схемы расчетов;
  - `<domain_id>` — идентификатор домена, для объекта которого нужно выполнить расчет;
  - `<entity_id>` — идентификатор сущности в домене, для объекта которой нужно выполнить расчет;
  - `<id>` — идентификатор объекта, для которого нужно выполнить расчет;
  - `<cache_first>` — логическая переменная: если установить значение `true`, то расчет не будет выполнен заново, если его результат будет найден в кэше; если установить значение `false`, то расчет будет выполнен заново в любом случае.

Примеры выполнения обобщающего расчета

▼ *Пример скрипта с `execSummary` в TypeScript*

```
id: exec_summary_calculation_bun
name: Exec summary calculation bun
description: Выполнение обобщающего расчета скриптом Bun (TypeScript) с помощью
метода execSummary
author: John Doe
tags: [bun, typescript, calculation]
version: 0.0.1
type: script
source: !bun |
  import * as sdk from '@playbooks/sdk';
  export async function main() {
    const command = {
      summaryCalculationSchemaId: "calculate_test",
      summaryCalculationId: "test_summary",
      calculations: [
        {
          calculationSchemaId: "calculate_test",
          calculationId: "test",
          object: {
            domainId: "evo.entity.category.standards",
            entityId: "Requirement",
            id: "551471a6-8bd2-4af7-a7ad-7af5432c9bea"
          },
          cacheFirst: false
        },
        {
          calculationSchemaId: "calculate_test",
          calculationId: "test",
          object: {
            domainId: "evo.entity.category.standards",
            entityId: "Requirement",
            id: "2395982b-0e28-4236-ab3d-3c84a29cb97c"
          },
```

```
          cacheFirst: false
        }
      ]
    }

    const result = await sdk.calculation.execSummary(command);
    console.log('result', result);
    return result;
  }
```

Плейбуки

В данном разделе представлен обзор структуры, принципов работы, процесса создания и выполнения плейбуков в системе. Управление плейбуками в системе осуществляется в разделе **Экспертиза**.

О плейбуках

Плейбуки — это наборы связанных сценариев, определяющих последовательность выполнения различных действий в системе.

Работа с плейбуком

Доступные операции над плейбуком:

- Создание плейбука
- Запуск плейбука
- Изменение плейбука
- Удаление плейбука

Создание плейбука

Плейбук можно создать самостоятельно или импортировать в систему.

Чтобы создать плейбук:

1. Перейдите в раздел **Экспертиза**. Система отобразит список элементов экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Плейбук**.
3. Создайте плейбук, определив логику его работы.

   Плейбук создается в файле формата YAML. Система отображает шаблон с полями, которые нужно заполнить для описания работы плейбука.

4. Нажмите на кнопку **Опубликовать версию**, чтобы сохранить изменения и опубликовать созданный плейбук. Система отобразит соответствующее уведомление. Новый плейбук появится в таблице раздела **Экспертиза**.

Чтобы плейбук стал доступен для запуска, его необходимо включить.

Вы можете создать плейбук в виде черновика без публикации с помощью кнопки **Сохранить черновик**. Система отобразит соответствующее уведомление. Черновик появится в таблице раздела **Экспертиза**.

Запуск плейбука

После того как плейбук протестирован и сохранен, он готов к запуску.

Чтобы запустить плейбук:

1. Перейдите в раздел **Экспертиза**. Система отобразит список элементов экспертизы.
2. Нажмите на строку плейбука в списке. Система отобразит в правой части экрана карточку этого плейбука с подробной информацией о нем.
3. Убедитесь, что плейбук включен. Переключатель в карточке плейбука должен быть переведен в активное положение.
4. Нажмите на кнопку **Запустить** в карточке плейбука. Плейбук будет запущен.

Плейбук можно также запустить из раздела **Запуски**.

Процесс выполнения плейбука в системе

1. Если у плейбука есть аргументы, после запуска плейбука отображается окно для ввода аргументов. Типы аргументов совпадают с типами переменных и зависят от языка плейбука.
2. После задания аргументов система преобразует их в формат JSON и передает плейбук на выполнение в воркер. При запуске плейбука система назначает ему специальный тег воркера. Нельзя смешивать этот тег с тегами, используемыми в **Экспертизе** для группировки плейбуков по бизнес-модели. Тег воркера, назначаемый плейбуку системой, указывает, какой конкретный воркер будет выполнять этот плейбук.

   С помощью менеджера воркеров можно назначать определенным воркерам плейбуки для выполнения, например, по языкам, на которых они написаны, или по компонентам системы.

3. После получения плейбука воркер находит исходный код плейбука в сервисе экспертизы по идентификатору плейбука, анализирует его и преобразует в готовую для выполнения структуру.

   Плейбук может включать различные виды модулей.

4. Для каждого отдельного модуля плейбука воркер создает отдельную задачу и выполняет следующие шаги:
   a. Если необходимо отдельно установить определенные зависимости модуля, воркер производит установку зависимостей в рамках отдельной задачи. Установленные зависимости в дальнейшем хранятся на этом воркере. Они используются при последующих выполнениях модуля на этом воркере, что сокращает время работы.
   b. После проверки наличия и установки зависимостей воркер проверяет, запускался ли этот модуль ранее с такими же параметрами и сохранились ли результаты этого запуска в кэше.

Если такие данные сохранены в кэше, воркер предоставит их, что ускорит выполнение процесса.

▼ *Пример*

Модуль производит следующую трансформацию входных данных в выходные данные — принимает объемные входные данные, преобразует их в JSON-файл, ищет определенное поле и передает его на выход. Если такой модуль с такими параметрами уже ранее запускался, воркер попытается найти кэшированные данные и предоставить их, не тратя времени на запуск скрипта.

c. Если готовый результат не найден, задача передается на выполнение. Каждый воркер в отдельный момент времени выполняет только одну задачу.

d. Воркер запускает процесс с интерпретатором требуемого языка. В файловой системе в отдельной директории сохраняются все необходимые данные. Например, аргументы, ссылки на зависимости, необходимые для запуска и указанные пользователем переменные окружения.

e. Система помещает модуль в специальную оболочку, позволяющую через библиотеку **playbooks_sdk** бесшовно получать доступ к переменным, элементам экспертизы, запускам и интеграциям с другими сервисами.

f. Модуль выполняется.

5. После выполнения всех модулей воркер завершает процесс, кэширует результаты и удаляет директорию с артефактами работы, так как все необходимые данные уже сохранены. После этого воркер переходит в режим ожидания следующей задачи.

Становятся доступными результат выполнения плейбука и логи. В зависимости от результата выполнения плейбука статус его запуска изменяется на **Выполнен** или **Ошибка**.

Изменение плейбука

Чтобы изменить плейбук:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку плейбука в списке. Система отобразит в правой части экрана карточку с подробной информацией о плейбуке.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек плейбука.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

4. Внесите требуемые изменения в конфигурацию плейбука.

5. Нажмите на кнопку **Обновить версию**. Новая версия плейбука с измененной конфигурацией будет опубликована. Система отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную конфигурацию плейбука в виде черновика без публикации помощью кнопки **Сохранить черновик**. Конфигурация будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки плейбука будет отображаться предупреждение о наличии неопубликованных изменений.

Удаление плейбука

Чтобы удалить плейбук:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку плейбука в списке. Система отобразит в правой части экрана карточку с подробной информацией о плейбуке.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**. Отобразится окно подтверждения удаления плейбука.
4. Нажмите на кнопку **Удалить**. Система удалит плейбук и отобразит соответствующее уведомление. Удаленный плейбук будет исключен из списка элементов экспертизы.

Вы также можете удалить группу элементов экспертизы. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные элементы и отобразит соответствующее уведомление. Удаленные элементы будет исключены из списка элементов экспертизы.

Структура RObject-схемы плейбука

Описание плейбука начинается с блока метаданных плейбука.

Содержимое плейбука находится в поле `body` — входные аргументы в поле `input` и модули в поле `modules`.

Блок настроек выполнения плейбука является необязательным.

Если поля, указанные в нем, заполнены, они являются значениями по умолчанию для соответствующих параметров самой задачи.

Поля метаданных

Запись метаданных включает следующие обязательные поля:

Поле	Описание	Тип данных
`id`	Уникальный идентификатор плейбука	строка
`type`	Тип элемента экспертизы. Для плейбука здесь всегда устанавливается значение `playbook`.	
`name`	Название. Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`), точка, подчеркивание и дефис без пробелов.	строка
`version`	Версия плейбука в формате Semantic Versioning.	строка

Вы также можете добавить в метаданные любые другие поля, содержащие дополнительные сведения о плейбуке (например, поле `author`).

Поле `input`

Входные аргументы плейбука задаются в поле `input`. После запуска плейбука система отобразит поля, описанные в поле `input` и заполненные значениями по умолчанию.

Структура поля `input` включает следующие поля:

Поле	Описание	Тип данных
`properties`	Массив записей с описанием входных аргументов	массив

Поле `properties`

Структура поля `properties` включает описание полей входных аргументов:

Поле	Описание	Тип данных
`type`	Тип поля	строка, число
`default`	Значение в поле, отображаемое по умолчанию	зависит от типа поля
`description`	Описание поля (отображается рядом с полем)	строка
`required`	Поле обязательно для заполнения. Значение обязательно должно быть передано.	логическое

Поле `modules`

Поле `modules` содержит массив модулей. Общая структура каждого модуля включает следующие

поля:

Поле	Описание	Тип данных	Обязательное поле
`id`	Идентификатор модуля	строка	да
`body`	Описание всех модулей, включенных в этот модуль, в заданной последовательности выполнения. Модуль может включать любое количество модулей разных типов. Поля модулей подробно описаны в статье Модули плейбуков.	массив	да
`failure_module`	Описание модуля сбоя, запускаемого в случае ошибки при выполнении модулей, описанных в поле `body`. Если в поле не задано описание модуля `failure_module`, в случае ошибки при выполнении модулей плейбука будет возвращено сообщение об ошибке. Модуль `failure_module` имеет такую же структуру, как и другие модули, задаваемые в плейбуках.	массив	нет
`summary`	Описание модуля. Может содержать необходимую пользователю информацию о работе и назначении модуля.	строка	нет
`sleep_secs`	Время ожидания перед началом выполнения модуля в секундах. Может использоваться, если по результатам выполнения предыдущего модуля система должна выполнить определенные действия (например, обновления). Позволяет задать время ожидания этого выполнения. См. описание поля.	массив	нет

Поле	Описание	Тип данных	Обязательное поле
`continue_on_error`	Продолжение выполнения модуля при ошибке. Позволяет определить, будет ли продолжено выполнение модуля при возникновении ошибки. Доступные значения: • `true` — продолжить выполнение модуля при ошибке; • `false` — остановить выполнение модуля.	логическое	нет
`retry`	Повторы выполнения модуля при его неуспешном выполнении. Позволяет настроить жестко или гибко заданное количество повторов выполнения модуля.	массив	нет
`priority`	Приоритетность выполнения модуля. В зависимости от важности задачи, выполняемой модулем, можно определить его приоритетность по сравнению с обработкой других элементов экспертизы. Возможные значения: от `1` (минимальная приоритетность) до `100` (максимальная приоритетность).	целое число	нет
`mock`	Отмена выполнения модуля. Может использоваться при отладке плейбука. Если в поле указано значение, оно будет возвращено вместо фактического выполнения модуля. См. описание поля.	строка	нет
`suspend`	Действие, выполнение которого должно предшествовать запуску модуля.		нет

Поле	Описание	Тип данных	Обязательное поле
`cache_ttl`	Время жизни (TTL, time to live) кэша данных скрипта в секундах. Может использоваться при работе модуля с внешним объектом (например, активными списками), чтобы соотносить время жизни кэша данных модуля и объекта. Параметр позволяет не хранить данные в кэше модуля дольше, чем они хранятся в кэше объекта. Время хранения определяется этим параметром. Если время жизни кэша данных модуля не указано в самой задаче, система берет его значение из этого поля.	целое число	нет
`timeout`	Максимальное время выполнения модуля в секундах. Если время выполнения модуля превышает максимальное, модуль будет принудительно остановлен с ошибкой выполнения.	целое число	нет
`delete_after_use`	Удаление значения, возвращаемого модулем после выполнения. Возможные значения: • `true` — значение, возвращаемое модулем, будет удалено; • `false` — значение, возвращаемое модулем, будет сохранено.	логическое	нет
`stop_after_if`	Остановка выполнения всех последующих модулей, если выражение, заданное в поле, возвращает значение `true`. См. описание поля.	логическое	нет

Поле `sleep_secs`

Структура поля `sleep_secs` включает следующие поля:

Поле	Описание
`type`	Формат времени ожидания. Может иметь одно из следующих значений: • `static` — статический формат; • `js` — формат выражения на языке JavaScript.
`value`	Значение времени ожидания. • Если выбран тип `static`, необходимо задать количество секунд ожидания: `value: <целое число>`. • Если выбран тип `js`, необходимо задать выражение на языке JavaScript. После выполнения выражение возвращает количество секунд ожидания: `value: <выражение на языке JavaScript>`.

▼ *Пример настройки в статическом формате, задающей 10-секундное ожидание перед началом выполнения модуля.*

```
# Настройка времени ожидания перед началом выполнения модуля
sleep_secs:
    # Статический формат
    type: static
    # Количество секунд ожидания
    value: 10
```

▼ *Пример настройки в формате выражения JavaScript. Указанное выражение возвращает время ожидания перед началом выполнения плейбука.*

```
# Настройка времени ожидания перед началом выполнения модуля.
sleep_secs:
    # Формат выражения JS.
    type: js
    # Выражение JS.
    value: flow_input.sleep + 100
```

Поле `retry`

Структура поля retry включает поля `constant` и `exponential`, определяющие условия повтора выполнения модуля в различных форматах.

Можно определить одно или оба условия. Если определены оба условия, сначала будет выполнено условие, заданное в поле `constant`.

Поле	Описание	Тип данных
`constant`	Жестко заданный формат. Может использоваться, если внешние сервисы, к которым обращается модуль, доступны, но могут иногда возвращать ошибки в ответ на запросы. В таких случаях лучше задавать настройку повторных запросов через определенный фиксированный интервал. Если выбран этот формат, необходимо заполнить следующие поля: • `attempts` — количество повторных попыток выполнения модуля; • `delay_secs` — интервал между попытками выполнения в секундах.	массив
`exponential`	Экспоненциальный формат. Может использоваться, если внешние сервисы, к которым обращается модуль, зависят от сетевого соединения. В таких случаях лучше выполнять повторные запросы через постоянно увеличивающийся интервал времени, ожидая появления сервиса в сети. Если выбран этот формат, необходимо заполнить следующие поля: • `attempts` — количество повторных попыток выполнения модуля; • `multiplier` — множитель, определяющий фактический интервал между попытками выполнения; • `delay_secs` — базовый интервал между попытками выполнения в секундах.	целое число или выражение

▼ *Пример настройки в жестко заданном формате*

Настройка задает 45 попыток повторного запуска и 67-секундный интервал между попытками повторного запуска.

```
# Настройка повторов выполнения модуля при его неуспешном выполнении.
retry:
  # Жестко заданный формат.
  constant:
     # Количество попыток повторного запуска
     attempts: 45
     # Интервал между попытками
     delay_secs: 67
```

▼ *Пример настройки в экспоненциальном формате*

Согласно этой настройке первая повторная попытка будет выполнена через 4 сек., вторая — через 12 сек. (4*3), а третья — через 36 сек. (12*3).

```
# Настройка повторов выполнения модуля при его неуспешном выполнении
retry:
   # Экспоненциальный формат
   exponential
      # Количество попыток повторного запуска.
      attempts: 3
      # Множитель, определяющий фактический интервал между попытками выполнения.
      multiplier: 3
      # Базовый интервал между попытками выполнения (сек.).
      delay_secs: 4
```

Поле `mock`

Структура поля `mock` включает следующие поля:

Поле	Описание
`new_value`	Поле для указания значения, которое будет возвращено системой вместо фактического выполнения модуля

▼ *Пример настройки*

Согласно этой настройке вместо выполнения модуля будет возвращено значение 1.

```
# Настройка, позволяющая отменить выполнение модуля.
mock:
    # Если указано это поле, модуль вернет указанное значение вместо выполнения.
    new_value: 1
```

Поле `stop_after_if`

Структура поля `stop_after_if` включает действия, определяющие, будет ли плейбук выполняться дальше.

Необходимо выбрать одно из действий.

Поле	Описание	Тип данных
`expr`	Выражение возвращает значение, определяющее, будет ли остановлено выполнение процесса (цикла или плейбука). Возможные значения: • Если поле используется в цикле: ◦ `true` — выполнение цикла будет остановлено; ◦ `false` — выполнение цикла не будет остановлено. • Если поле используется в модуле, который не является циклом: ◦ `true` — выполнение плейбука будет остановлено; ◦ `false` — выполнение плейбука не будет остановлено.	логическое
`skip_if_stopped`	Позволяет пометить остановленный плейбук как пропущенный. Возможные значения: • `true` — остановленный плейбук или цикл будет помечен как пропущенный; • `false` — остановленный плейбук или цикл будет помечен как выполненный.	логическое

Блок настроек выполнения плейбука

Блок включает следующие поля:

Поле	Описание	Тип данных
`same_worker`	Необходимость выполнения родительского плейбука и текущего плейбука на одном и том же воркере. Используется, если текущий плейбук запускается в ходе выполнения определенного (родительского) плейбука. Если в родительском плейбуке не заполнено поле `same_worker`, именно это поле будет определять, выполнится ли текущий плейбук на том же воркере, что и родительский. Возможные значения: • `true` — оба плейбука выполняются на одном и том же воркере; • `false` — текущий и родительский плейбуки могут выполняться на разных воркерах.	булево

Поле	Описание	Тип данных
`concurrent_limit`	Количество параллельных выполнений плейбука. Используется, если для корректной работы плейбука необходимо ввести это ограничение. Например, у сервера, к которому обращается задача при выполнении плейбука, может быть ограниченное количество входных соединений. Этот параметр позволяет ограничить количество выполняющихся экземпляров плейбука количеством входных соединений. В результате будет установлено разрешенное количество соединений для выполнений плейбука, а лимит превышен не будет.	целое число
`concurrency_time_window_t`	Максимально допустимое время выполнения задачи в секундах при параллельном выполнении. Если оно превысит значение, указанное в этом поле, задача будет отменена.	целое число
`skip_expr`	Необходимость выполнения плейбука по результатам проверки определенного условия. Выражение на языке JavaScript. Выполняется до запуска плейбука, проверяет определенные условия и по результатам проверки определяет необходимость его запуска. Пример. В ходе выполнения плейбука предполагается его обращение к определенному сервису. Выражение может проверить доступность порта сервиса через утилиту `ping`. Если порт недоступен, плейбук выполнен не будет. Возвращаемое значение выражения: • `true` — плейбук не будет выполнен; • `false` — плейбук будет выполнен.	строка
`cache_ttl`	Время жизни (TTL, time to live) кэша данных плейбука в секундах. Параметр можно использовать при работе плейбука с внешним объектом (например, активными списками). Он позволяет не хранить данные в кэше плейбука дольше, чем они хранятся в кэше объекта. Время хранения кэша данных плейбука определяется этим параметром.	целое число

Поле	Описание	Тип данных
`early_return`	Необходимость повторного запуска плейбука в случае ошибки. Возможные значения: • `true` — в случае сбоя выполнения плейбука система не пытается запустить плейбук повторно; • `false` — в случае сбоя выполнения плейбука система предпринимает попытку повторного запуска.	логическое
`priority`	Приоритетность выполнения плейбука. В зависимости от важности задачи, выполняемой плейбуком, можно определить его приоритетность по сравнению с обработкой других элементов экспертизы. Возможные значения: от `1` (минимальная приоритетность) до `100` (максимальная приоритетность).	целое число

Составление плейбука

Для создания плейбука:

1. Опишите метаданные плейбука (`id`, `type`, `name`, `version`).

   ▼ *Пример заполненной структуры метаданных*

   ```
   id: playbook_new_version
   type: playbook
   name: playbook_new_version_name
   version: 1.0.0
   ```

2. Опишите содержимое плейбука в поле `body`:

   a. Опишите блок входных аргументов в поле `input`.

   ▼ *Пример записей в поле `input`*

   ```
   input:
       # Блок, в котором перечисляются входные аргументы.
       properties:
          # Первый входной аргумент.
          employeename:
               # Тип первого входного аргумента.
               type: string
               # Описание первого входного аргумента.
               description: "Employee first name"
               # Значение первого входного аргумента по умолчанию.
               default: "default_employeename"
               # Аргумент является обязательным.
   ```

```
            required: true
        # Второй входной аргумент.
        age:
            type: number
            default: 35
            description: "Age description"
```

b. Опишите в поле `modules` модули требуемых типов. При запуске плейбука все описанные модули выполняются последовательно.

▼ *Структура поля* `modules` *плейбука:*

```
modules:
    # Идентификатор первого модуля плейбука.
    - id:
      # Поле, содержащее описание первого модуля (скрипты, плейбуки, циклы со
своими модулями и т.д.).
      body:
        # Тип первого модуля.
        type:
        # Под полем type располагаются записи полей, описывающие модуль
выбранного типа. Набоор полей зависит от типа модуля.

    # Идентификатор второго модуля плейбука.
    - id:
    # Поле, содержащее описание второго модуля.
    body:
       # Тип второго модуля
       type:
    # И так далее.
    ...
```

3. Добавьте описание модуля сбоя `failure_module` для плейбука (опционально). Модуль сбоя имеет такую же структуру, что и другие модули.
4. Добавьте блок настроек выполнения плейбука (опционально).

Пример плейбука

```
id: playbook_with_rawscript
name: Playbook with rawscript
description: Плейбук с модулем rawscript
version: 0.0.1
type: playbook
body:
  input:
    properties:
      name_string:
        type: string
        description: ""
```

```
      default: "1se"
  modules:
      - id: rawscript_module_x
        body:
          type: rawscript
          input:
            x: # Значение аргумента берется из name_string
              type: js
              expr: playbook_input.name_string
          source: !py |
            def main(x: str):
              new_text = 'Текст для вывода ' + x
              return(new_text)
          retry:
            constant:
              attempts: 2
              delay_secs: 5
      - id: rawscript_module_y
        body:
          type: rawscript
          input:
            y: # Значение аргумента берется из name_string
              type: js
              expr: playbook_input.name_string
          source: !bun |
            function main(y: string): void {
              const newText = 'Текст для вывода ' + y;
              console.log(newText);
            }
          retry:
            constant:
              attempts: 2
              delay_secs: 5
```

Особенности работы с модулями

Цикл

Модуль цикла содержит другие модули и настройки, определяющие количество повторов их выполнения в определенном порядке. Воркер выполняет все модули, содержащиеся в цикле в соответствии с настройками цикла. На каждое повторное выполнение модуля в рамках цикла создается по крайней мере одна отдельная задача. Количество задач может быть больше, так как установка зависимостей требует создания отдельной задачи.

Ветвления

Модуль ветвлений содержит другие модули и настройки, определяющие, при каких условиях модули будут выполняться. На каждое ветвление в рамках модуля ветвления создается по крайней мере одна отдельная задача. Количество задач может быть больше, так как установка зависимостей требует создания отдельной задачи.

Количество задач может быть больше, так как установка зависимостей требует создания отдельной задачи.

Повторы запуска модулей и обработка ошибок

Если какой-либо модуль в составе плейбука выполняется с ошибкой, его можно перезапустить указанное количество раз с указанным интервалом, используя настройки расписания в параметрах запуска.

Если модуль запустить все же не удалось, плейбук попытается обработать ошибку с помощью специального модуля обработки ошибок.

Доступ к хранилищу данных

Когда плейбук запускается, система создает в специальном хранилище данных запись о запуске. Эти записи можно использовать в плейбуках, например, как условие срабатывания определенного правила. Каждая запись в хранилище имеет свой идентификатор (ID интеграции), на который пользователь может ссылаться в плейбуках.

Модули плейбуков

Описание модулей содержится в поле `modules` плейбука.

Содержание полей зависит от типа модуля. В системе существуют следующие типы модулей:

- script
- playbook
- forloop
- case
- all
- rawscript
- identity
- whileloop
- approval

Специальные переменные

При работе с плейбуками можно использовать специальные переменные.

Тип `script`

Описание скрипта включает следующие поля:

Поле	Описание	Тип данных
`type`	Тип модуля. Устанавливается значение `script`.	строка
`id`	Ссылка на идентификатор существующего скрипта, который будет выполняться на этом шаге плейбука	строка

Поле	Описание	Тип данных
`input`	Входные аргументы	массив
`tag_override`	Переопределение тега скрипта. От значения тега зависит, на каком воркере будет выполняться скрипт. Поле используется, если скрипт в рамках этого плейбука должен выполняться не на том воркере, где обычно выполняется скрипт, а на каком-либо другом.	строка

▼ *Пример описания модуля типа `script` в поле `modules` в схеме плейбука:*

```
modules:
    # Идентификатор модуля
    - id: "1_module_id"
      Группа полей, описывающая скрипт
      body:
        # Тип модуля - скрипт
        type: script
        # Идентификатор скрипта
        id: 84895c89-468e-41f8-9f52-ad854d36734f
        # Входные аргументы скрипта
        input:
            age:
                type: static
                expr: 'arg_value'
            name:
                type: js
                expr: flow_input.filename
        # Переопределение тега скрипта
        tag_override: 'worker_456'
```

Тип `playbook`

Описание плейбука включает следующие поля:

Поле	Описание	Тип данных
`type`	Тип модуля. Устанавливается значение `playbook`.	строка
`input`	Входные аргументы	массив

▼ *Пример описания модуля типа `playbook` в поле `modules` в схеме плейбука:*

```
modules:
    # Идентификатор модуля
    - id: "2_module_id"
      # Группа полей, описывающая плейбук
      body:
        # Тип модуля - playbook
```

```
type: playbook
id: 893495c89-434e-49s8
# Входные аргументы плейбука
input:
   age:
        type: js
        expr: flow_input_age
    name:
        type: js
        expr: flow_input.filename
```

Тип `forloop`

Описание типа модуля `forloop` (цикл) включает следующие поля:

Поле	Описание	Тип данных
`type`	Тип модуля. Устанавливается значение `forloop`.	строка
`modules`	Массив описаний модулей, включенных в текущий цикл. См. описание поля.	массив
`skip_failures`	Возобновление выполнения цикла при ошибке. Доступные значения: • `true` — если при выполнении какого-либо модуля в цикле возникнет ошибка, выполнение цикла начнется повторно, с новой итерации. • `false` — если при выполнении какого-либо модуля в цикле возникнет ошибка, выполнение цикла будет прервано.	логическое
`parallel`	Параллельное выполнение итераций цикла. Доступные значения: • `true` — итерации цикла могут выполняться параллельно. • `false` — параллельное выполнение итераций цикла запрещено.	булево
`parallelism`	Количество параллельных выполнений итераций цикла	целое число
`iterator`	Параметры итератора. См. описание поля.	массив

Поле `modules`

В поле `modules` задается описание модулей, выполняемых в рамках этого цикла.

Структура каждого модуля включает обязательные поля `id` и `body`, а также другие поля, набор которых зависит от типа модуля.

Структура поля включает следующие поля:

Поле	Описание	Тип данных
`id`	Идентификатор	строка
`body`	Массив записей, описывающих модули в цикле	массив

Поле `iterator`

В поле `iterator` задается описание настроек итератора. Итератор определяет количество итераций цикла и входные аргументы для каждой итерации.

Структура поля включает следующие поля:

Поле	Описание	Тип данных
`type`	Тип формата данных итератора. Может иметь одно из следующих значений: • `static` — статичный формат; • `js` — формат JavaScript.	строка
`value` или `expr`	Настройка количества итераций и входных аргументов итераций цикла. • Если выбран тип `static`, в поле `value` необходимо задать численные значения аргументов для каждой итерации `value: [число1, число2, ⋯ ]`. • Если выбран тип `js`, в поле `expr` необходимо задать выражение на языке JavaScript. Выполненное выражение возвращает значения аргументов для каждой итерации: `expr: [выражение1, выражение2, ⋯ ]`. Количество аргументов в полях `value` или `expr` определяет количество итераций в цикле.	число или выражение

▼ *Пример описания модуля типа `forloop` в поле `modules` в схеме плейбука:*

Цикл включает в себя два модуля. Для цикла разрешено параллельное выполнение пяти итераций.

В настройках итератора указано количество итераций (8) и входные аргументы для каждой итерации.

```
modules:
    # Идентификатор модуля цикла
    - id: "module_id"
    # Группа полей, описывающая цикл
      body:
        # Тип модуля - forloop.
        type: forloop
        # Блок описания модулей в составе цикла.
```

```
        modules:
            # Идентификатор первого модуля в составе цикла.
            - id: "2_1_module_id"
            # Группа полей, описывающая первый модуль.
              body:
                type: rawscript
                source: !py
                    def main(x):
                        return x
                input:
                    x:
                        type: js
                        expr: flow_input.iter.value

           # Идентификатор второго модуля в составе цикла.
           - id: "2_2_module_id"
              # Группа полей, описывающая второй модуль.
              body:
                type: identity
        # Параметры итератора
        iterator:
            # Тип формата данных итератора (статический).
            type: static
            # Выражение, задающее входные аргументы для каждой итерации цикла.
Количество аргументов - 8, следовательно, цикл имеет 8 итераций.
            expr: "[2, 4, 6, 8, 10, 12, 14, 16]"
        # Разрешено параллельное выполнение итераций.
        parallel: true
        # Максимальное количество параллельных итераций.
        parallelism: 5
        # Если при выполнении какого-либо модуля в цикле возникнет ошибка,
выполнение цикла начнется повторно, с новой итерации.
        skip_failures: true
```

Тип `case`

В этом типе задается конкурентное выполнение нескольких веток с добавлением описания модуля (модулей), запускаемых по умолчанию, то есть, если ни одно из приведенных ниже условий не выполняется.

Описание типа модуля case (ветвление) включает следующие поля:

Поле	Описание	Тип данных
`type`	Тип модуля. Устанавливается значение `case`.	строка
`default`	См. описание поля.	массив
`branches`	См. описание поля.	массив

Поле `default`

В поле `default` задается описание модуля или модулей, выполняемых по умолчанию.

Структура каждого модуля включает обязательные поля `id` и `body`, а также другие требуемые поля.

Поле	Описание	Тип данных
`id`	Идентификатор модуля. Для цикла устанавливается значение `forloop`.	строка
`body`	Описание модуля. Набор полей зависит от типа модуля.	

Поле `branches`

В поле `branches` задается описание модуля или модулей, запускаемых при выполнении определенных заданных условий.

Структура поля включает следующие поля:

Поле	Описание	Тип данных
`summary`	Описание ветки. Необязательное поле.	строка
`condition`	Условие выполнения. Указывается переменная и ее допустимые значения для выполнения условия.	
`modules`	Массив описаний модулей, запускаемых при выполнении условия	

▼ *Пример описания модуля типа case в поле `modules` в схеме плейбука:*

Модуль включает в себя описание двух модулей, запускаемых по умолчанию, а также два условия. При выполнении какого-либо условия запускается модуль, описанный под этим условием.

```
modules:
    # Идентификатор модуля case
    - id: 1_module
      # Группа полей, описывающая ветвление
      body:
         # Тип модуля - case
         type: case
         # Блок, описывающий модули, запускаемые по умолчанию (если ни одно условие
не выполнено)
         default:
           - id: 1_1_module
             body:
               type: script
               id: 64fe191d-3319-47ea-a87b-046c8df4032b
               input: {}
           - id: 1_2_module
             body:
                type: script
```

```
                id: 93fe803d-7375-456a-a28b-6c84hj73j02c
                input: {}
      # Блок, описывающий модули, запускаемые при выполнении указанных условий
      branches:
          # Необязательное описание первого ветвления
          - summary: branch_1
            # Условие первого ветвления
            condition: flow_input.age < 50
            # В поле modules размещается описание модулей, запускаемых при
выполнении условия первого ветвления
            modules:
              - id: 1_3_module
                 body:
                    type: script
                    id: 66fa731d-9378-25wa-a93q-003c8df9031a
                    input: {}
         # Необязательное описание второго ветвления
         - summary: branch_2
           # Условие второго ветвления
           condition: flow_input.age < 100
           # В поле modules размещается описание модулей, запускаемых при выполнении
условия второго ветвления
           modules:
               - id: 1_4_module
                 body:
                    type: script
                    id: 191dr623rf-9058-99ea-a23b-904c8df83564w
                    input: {}
```

Тип `all`

В этом типе задается параллельное выполнение веток в плейбуке.

Описание типа модуля `all` включает следующие поля:

Поле	Описание	Тип данных
`type`	Тип модуля. Устанавливается значение `all`.	строка
`branches`	См. описание поля.	массив
`parallel`	Параллельное выполнение веток. Доступные значения: • `true` — ветки могут выполняться параллельно. • `false` — параллельное выполнение веток запрещено. Ветки будут выполняться последовательно.	булево

Поле `branches`

В поле `branches` задается описание веток с модулями, запускаемыми для параллельного

выполнения.

Структура поля включает следующие поля:

Поле	Описание	Тип данных
`summary`	Описание ветки. Необязательное поле.	строка
`parallel`	Параллельное выполнение модулей в ветке. Доступные значения: • `true` — модули в ветке могут выполняться параллельно. • `false` — параллельное выполнение модулей в ветке запрещено. Модули будут выполняться последовательно.	булево
`modules`	Массив описаний модулей в ветке	массив

▼ *Пример описания модуля типа* `all` *в поле* `modules` *в схеме плейбука.*

Модуль включает в себя описание двух веток, содержащих описания модулей. Ветки могут запускаться параллельно.

```
modules:
    # Идентификатор модуля case
    modules:
     - id: 1_module
       body:
         type: all
         # Блок, описывающий модули, запускаемые в ветках
         branches:
              # Описание первой ветки
              - summary: branch_1
                # Запрет параллельного запуска модулей в ветке
                parallel: false
                modules:
                  - id: 1_1_module
                    body:
                      type: script
                      id: 64fe191d-3319-47ea-a87b-046c8df4032b
                      input: {}
              # Описание второй ветки
              - summary: branch_2
                # Запрет параллельного запуска модулей в ветке
                parallel: false
                modules:
                 - id: 1_2_module
                   body:
                    type: script
                    id: 191dr623rf-9058-99ea-a23b-904c8f83564w
    input: {}
    # Ветки будут запущены параллельно
```

```
    parallel: true
```

Тип `rawscript`

Тип модуля `rawscript` позволяет протестировать созданный и неопубликованный скрипт перед публикацией.

Описание включает следующие поля:

Поле	Описание	Тип данных
`type`	Тип модуля. Устанавливается значение `rawscript`.	строка
`source`	Выполняемый код скрипта	
`input`	Входные аргументы для выполнения кода скрипта	массив
`tag`	Тег скрипта	строка

Поле `source`

Значением поля должен быть тег, соответствующий одному из поддерживаемых языков:

- `!bun` — TypeScript, среда выполнения Bun;
- `!deno` — TypeScript, среда выполнения Deno;
- `!py` — Python;
- `!go` — Go;
- `!pwsh` — PowerShell;
- `!bash` — Bash.

Ниже в поле необходимо ввести код, который будет протестирован.

▼ *Пример описания модуля типа `RawScript` в поле modules в схеме плейбука*

```
modules:
    # Идентификатор модуля
    - id: "1_module_id"
      # Группа полей, описывающая неопубликованный скрипт
      body:
        # Тип модуля - rawscript.
        type: rawscript
        # Входные аргументы.
        input:
            arg1:
               type: static
               value: 1
            arg2:
               type: static
               value: qwerty
            arg3:
               type: js
```

```
            expr: flow_input.input_arg
      # Описание языка выполнения скрипта.
      source: !py
         # Выполняемый код.
         print(123)
```

Тип `identity`

Этот тип модуля не выполняет никаких действий и используется при отладке. Поступающие в модуль входные значения передаются на выход неизмененными.

Описание модуля типа `identity` включает следующие поля:

Поле	Описание	Тип данных
`type`	Тип модуля. Устанавливается значение `identity`.	строка

▼ *Пример описания модуля типа `identity` в поле `modules` в схеме плейбука*

```
modules:
    # Идентификатор модуля identity.
    - id: 1_module
      # Группа полей, описывающая модуль.
      body:
         # Тип модуля - identity.
         type: identity
```

Тип `whileloop`

Тип модуля `whileloop` используется для описания цикла (многократного повторения определенной последовательности модулей), выполняемого при соблюдении заданного условия.

Оно может применяться, например, если необходимо обработать неизвестное количество событий или сущностей, либо выполнить неизвестное количество шагов. Их точное количество определяется ограничениями, указанными в настройках поля `stop_after_if`, описанного в общей структуре модуля.

Возможные сценарии использования:

- Плейбук отправляет на обработку фрагменты какого-то массива данных. Цикл отправки будет выполняться, пока система не удостоверится, что переданы абсолютно все фрагменты.
- Плейбук передает на внешний сервис данные для обработки. Объем данных, который сервис сочтет достаточным для обработки, заранее неизвестен. Цикл будет прерван, когда от сервиса поступит информация о том, что данные переданы в достаточном количестве.

Структура модуля включает следующие поля:

Поле	Описание	Тип данных
`type`	Тип модуля. Устанавливается значение `whileloop`.	строка

Поле	Описание	Тип данных
`modules`	Описания модулей, включенных в цикл. См. описание поля.	объект
`skip_failures`	Возобновление выполнения цикла при ошибке. Доступные значения: • `true` — если при выполнении какого-либо модуля в цикле возникнет ошибка, выполнение цикла начнется повторно, с новой итерации. • `false` — если при выполнении какого-либо модуля в цикле возникнет ошибка, выполнение цикла будет прервано.	логическое

Поле `modules`

В поле `modules` задается описание модулей, выполняемых в рамках этого цикла.

Структура каждого модуля включает обязательные поля `id` и `body`, а также другие поля, набор которых зависит от типа модуля.

Структура поля включает следующие поля:

Поле	Описание	Тип данных
`id`	Идентификатор	строка
`body`	Тело модуля	объект

Сразу после описания модуля в схеме плейбука укажите обязательное поле `stop_after_if`, описанное в общей структуре модуля. Его наличие необходимо для правильной работы этого модуля.

▼ *Пример*

Описание модуля типа `forloop` в поле `modules` в схеме плейбука:

Цикл включает в себя два модуля.

В настройках поля `stop_after_if` указано, что остановленный модуль будет запущен повторно. Выражение вернет значение, которое определит, будут ли запускаться последующие модули.

```
modules:
   # Идентификатор модуля цикла.
  - id: while_loop
    body:
      type: whileloop
      # Возможность возобновления выполнения цикла при ошибке.
      skip_failures: false
      modules:
        # Идентификатор первого модуля в составе цикла.
          - id: "2_1_module_id"
```

```
            # Группа полей, описывающая первый модуль.
            body:
              type: rawscript
              source: !py
                  def main(x):
                      return x
              input:
                  x:
                      type: js
                      expr: flow_input.iter.value

        # Идентификатор второго модуля в составе цикла.
        - id: "2_2_module_id"
          # Группа полей, описывающая второй модуль.
          body:
            type: identity
    # Возможность запуска остановленного модуля и всех последующих модулей.
    stop_after_if:
      expr: result.num_arg == 3
      skip_if_stopped: false
```

Тип `approval`

Тип модуля `approval` переводит плейбук в режим ожидания до тех пор, пока пользователь вручную не согласует выполнение данного модуля. До момента согласования плейбук находится в режиме ожидания. Если время ожидания истекло, выполнение плейбука завершается со статусом **Истекло время согласования**.

Структура модуля включает следующие поля:

Поле	Описание	Тип данных
`timeout_sec`	Время ожидания одобрения в секундах. Если за указанный интервал времени запрос не был согласован, выполнение плейбука прекращается.	целое число
`message`	Сопроводительное сообщение, отображаемое в момент ожидания согласования	текст

Специальные переменные в плейбуках

- Переменная results
- Переменная previous_result
- Переменная relative_results
- Переменная iter_ctx
- Переменная playbook_args

Переменная results

Описание переменной `results` находится в статье Переменная results.

Переменная previous_result

Логика использования переменной `previous_result` такая же, как и для переменной `results`, однако здесь происходит автоматическая подстановка пути к необходимому результату. Переменная `previous_result` позволяет получить результаты предыдущего модуля, который выполнился в контексте того же корневого плейбука.

▼ *Пример использования переменной* `previous_result`

```
id: playbook_with_forloop_previous_result
name: Playbook with forloop previous_result
version: 0.0.10
type: playbook
tags: [playbook_special_envs]
body:
  input:
    properties:
      name:
        type: string
        description: ""
        default: ""
  modules:
    - id: test_forloop
      body:
        type: forloop
        modules:
          -   id: forloop_m_0
              body:
                type: all
                branches:
                  - summary: Branch 1
                    id: branch_1
                    modules:
                      - id: dummy_string_branch_1
                        body:
                            type: script
                            id: py_dummy_any_arg
                            input:
                                any_arg:
                                    type: js
                                    expr: playbook_args.name
                    parallel: false
                    skip_failure: true
                parallel: false
          -   id: forloop_m_1
              body:
                  type: script
```

```
                id: py_dummy_any_arg
                input:
                  any_arg:
                      type: js
                      expr: previous_result.branch_1
      iterator:
        type: js
        expr: "[2, 3]"
      parallel: false
```

▼ *Пример хранимого результата переменной* `previous_result`

```
dummy_string_branch_1:
We got any_arg: Naame

forloop_m_1_:
We got any_arg: {"dummy_string_branch_1":{"any_arg":"Naame"}}
```

Переменная relative_results

Логика использования переменной `relative_results` такая же, как и для переменной `results`, однако здесь происходит автоматическая подстановка пути к необходимому результату. Переменная `relative_results` автоматически начинает путь к результату от родителя модуля.

▼ *Пример использования переменной* `relative_results`

```
id: playbook_with_forloop_relative_results
name: Playbook with forloop relative results
version: 0.0.6
type: playbook
tags: [playbook_special_envs]
body:
  input:
    properties:
      name:
        type: string
        description: ""
        default: ""
  modules:
    - id: test_forloop
      body:
        type: forloop
        modules:
          -   id: forloop_m_0
              body:
                  type: script
                  id: py_dummy_string_arg
                  input:
                    string_arg:
                        type: js
```

```
                        expr: playbook_args.name
          -   id: forloop_m_1
              body:
                  type: script
                  id: py_dummy_any_arg
                  input:
                      any_arg:
                          type: js
                          expr: relative_results # parent is test_forloop[n]
          - id: for_loop_m_2
            body:
              type: case
              default:
                -   id: default_case_module
                    body:
                        type: script
                        id: py_dummy_string_arg
                        input:
                            string_arg:
                                type: js
                                expr: playbook_args.name
                -   id: default_case_module_2
                    body:
                        type: script
                        id: py_dummy_any_arg
                        input:
                            any_arg:
                                type: js
                                expr: relative_results.default_case_module # parent
is test_forloop[n].for_loop_m_2.default
              branches:
                -   summary: branch_1
                    id: conditional_case_module
                    condition: playbook_args.name == 'Naame'
                    modules:
                      -   id: cond_case_m_1
                          body:
                              type: script
                              id: py_dummy_string_arg
                              input:
                                  string_arg:
                                      type: js
                                      expr: playbook_args.name
                      -   id: cond_case_m_2
                          body:
                              type: script
                              id: py_dummy_any_arg
                              input:
                                  any_arg:
                                      type: js
                                      expr: relative_results.cond_case_m_1 # parent
```

```
is test_forloop[n].for_loop_m_2.conditional_case_module
        iterator:
          type: js
          expr: "[2]"
        parallel: false
    -   id: last_module
        body:
            type: script
            id: py_dummy_any_arg
            input:
                any_arg:
                    type: js
                    expr: relative_results # basically the same as results, becase
parent is root
    -   id: last_module_2
        body:
            type: script
            id: py_dummy_any_arg
            input:
                any_arg:
                    type: js
                    expr: relative_results.test_forloop[0].forloop_m_0 # basically
the same as results, becase parent is root
```

▼ *Пример хранимого результата переменной* `relative_results`

```
forloop_m_0:

We got string_arg: Naame

forloop_m_1:

We got any_arg: {"iter_idx":0,"iter_value":2,"forloop_m_0":{"string_arg":"Naame"}}

cond_case_m_1:

We got string_arg: Naame

cond_case_m_2:

We got any_arg: {"string_arg":"Naame"}

last_module # (находится в корне, поэтому по сути для него relative_results =
results):

{"test_forloop":[{"iter_idx":0,"iter_value":2,"forloop_m_0":{"string_arg":"Naame"},"
forloop_m_1":{"any_arg":"{\"iter_idx\":0,\"iter_value\":2,\"forloop_m_0\":{\"string_
arg\":\"Naame\"}}"},"for_loop_m_2":{"chosen_branch":"conditional_case_module","condi
tional_case_module":{"cond_case_m_1":{"string_arg":"Naame"},"cond_case_m_2":{"any_ar
g":"{\"string_arg\":\"Naame\"}"}}}}]}
```

```
last_module_2 # (находится в корне, поэтому по сути для него relative_results =
results):

We got any_arg: {"string_arg":"Naame"}
```

Переменная iter_ctx

Переменная `iter_ctx` содержит в себе информацию о текущей и предыдущей итерациях цикла:

```
iter_ctx: {"previous_iter":{"index":0,"value":2},"current_iter":{"index":1,"value":4}}
```

▼ *Пример использования переменной* `iter_ctx`

```
id: playbook_with_forloop_iter_ctx
name: Playbook with forloop iter ctx
version: 0.0.13
type: playbook
body:
  input:
    properties: {}
  modules:
    - id: test_forloop
      body:
        type: forloop
        modules:
          -   id: forloop_m_1
              body:
                  type: script
                  id: py_dummy_any_arg
                  input:
                      any_arg:
                          type: js
                          expr: iter_ctx
          -   id: forloop_m_2
              body:
                  type: script
                  id: py_dummy_any_arg
                  input:
                      any_arg:
                          type: js
                          expr: iter_ctx.current_iter.index
          -   id: forloop_m_3
              body:
                  type: script
                  id: py_dummy_any_arg
                  input:
                      any_arg:
                          type: js
```

```
                        expr: iter_ctx.previous_iter.value
        iterator:
          type: js
          expr: "[2, 4]"
        parallel: false
```

Использование переменной iter_ctx во вложенных итерациях

Переменная `iter_ctx` может использоваться также для того, чтобы узнать контекст родительской итерации. Она хранит в себе следующие данные:

- `current_iter (index, value)` — информация о текущей итерации.
- `previous_iter (index,value)` — информация о предыдущей итерации, если эта итерация существует.
- `parent_iter` — информация о родительской итерации, если эта итерация существует. Имеет такую же структуру, как и `iter_ctx`.

Если имеется дополнительное вложение, внутри `parent_iter` может храниться `parent_iter`. Количество вложений не ограничено.

▼ *Структура переменной* `iter_ctx`

```
iter_ctx: {
    "current_iter": {
        "index": 1,
        "value": 4
    },
    "previous_iter": {
        "index": 0,
        "value": 2
    },
    "parent_iter": {
        "previous_iter": {
            ...
        },
        "current_iter": {
            ...
        },
        "parent_iter": {
            ...
        }
    }
}
```

```
id: playbook_with_nested_iter_ctx
name: Playbook with nester iter ctx
version: 0.0.6
type: playbook
```

```
tags: [playbook_special_envs]
body:
  input:
    properties: {}
  modules:
    - id: root_forloop
      body:
        type: forloop
        modules:
          -   id: root_forloop_script
              body:
                  type: script
                  id: py_dummy_any_arg
                  input:
                      any_arg:
                          type: js
                          # Get whole iter_ctx.
                          expr: iter_ctx
          - id: root_forloop_nested_forloop
            body:
              type: forloop
              modules:
                -   id: root_forloop_nested_forloop_script
                    body:
                        type: script
                        id: py_dummy_any_arg
                        input:
                            any_arg:
                                type: js
                                # Get current iter of parent iter (root_forloop).
                                expr: iter_ctx.parent_iter.current_iter
                - id: root_forloop_nested_forloop_branch_all
                  body:
                    type: all
                    parallel: false
                    branches:
                      - summary: Branch 1
                        id: root_forloop_nested_forloop_branch_all_branch1
                        modules:
                          - id: root_forloop_nested_forloop_branch_all_branch1_script1
                            body:
                                type: script
                                id: py_dummy_any_arg
                                input:
                                    any_arg:
                                        type: js
                                        # Get current iter of parent iter
(root_forloop).
                                        expr: iter_ctx.parent_iter.current_iter
                          - id: root_forloop_nested_forloop_branch_all_branch1_script2
                            body:
```

```
                                type: script
                                id: py_dummy_any_arg
                                input:
                                    any_arg:
                                        type: js
                                        # Get whole iter ctx.
                                        expr: iter_ctx
                            - id:
root_forloop_nested_forloop_branch_all_branch1_nnforloop
                              body:
                                type: forloop
                                modules:
                                  - id:
root_forloop_nested_forloop_branch_all_branch1_nnforloop_script1
                                    body:
                                        type: script
                                        id: py_dummy_any_arg
                                        input:
                                            any_arg:
                                                type: js
                                                # Get index of self current iter
(root_forloop_nested_forloop_branch_all_branch1_nnforloop).
                                                expr: iter_ctx.current_iter.index
                                  - id:
root_forloop_nested_forloop_branch_all_branch1_nnforloop_script2
                                    body:
                                        type: script
                                        id: py_dummy_any_arg
                                        input:
                                            any_arg:
                                                type: js
                                                # Get current iter of parent of parent
(root_forloop).
                                                expr:
iter_ctx.parent_iter.parent_iter.current_iter
                                iterator:
                                  type: js
                                  expr: "[2]"
                                parallel: false
                                skip_failures: true
                iterator:
                  type: js
                  expr: "[2]"
                parallel: false
        iterator:
          type: js
          expr: "[2]"
        parallel: false
```

Переменная playbook_args

Переменная `playbook_args` представляет собой аргументы плейбука. Ознакомиться с ее использованием можно в вышеприведенных примерах.

Переменная results

Переменная `results` содержит результаты выполнения корневого плейбука в иерархическом виде. Если плейбук не является корневым, то есть он был запущен другим плейбуком, внутренний плейбук хранит только свои внутренние результаты.

Передача значения переменной во входные аргументы

Переменную `results` можно передавать во входные аргументы модулей.

▼ *Пример*

В этом примере передаются результаты модуля `forloop_m_1` первой итерации модуля `test_forloop`.

```
input:
    any_arg:
        type: js
        expr: results.test_forloop[0].forloop_m_1
```

При необходимости отладки можно передавать результаты в полном виде: `expr: results`.

Таблица ниже описывает, какой результат хранит переменная `results` в определенных модулях.

Модуль	Результат	Пример
Скрипт или плейбук	Результат выполнения конкретного запуска	
`forloop`	Массив объектов, где каждый объект содержит информацию о номере итерации, итерируемом значении и результатах выполнения внутренних модулей.	`[ { "iter_idx": 0, "iter_value": 2, "inner_for_loop_module": { "num_arg": 3 }, "inner_for_loop_module_2":{ "num_arg": 3 } ... }, ... ]`

Модуль	Результат	Пример

whileloop	Массив объектов, где каждый объект содержит информацию о номере итерации и результатах выполнения внутренних модулей	[
		  {
		  "iter_idx": 0,
		  "inner_while_loop_module": {
		  "num_arg": 3
		  },
		  "inner_while_loop_module_2": {
		  "num_arg": 3
		  }
		  ...
		  },
		...
		]

case	Объект, содержащий информацию о выбранной ветке и результатах выполнения внутренних модулей	{
		  "chosen_branch":
		"conditional_case_module",
		  "conditional_case_module": {
		  "Dummy module with 1 arg": {
		  "string_arg": "Naame"
		  }
		  }
		}

all	Иерархические результаты вида branch_id.branch_module_id	{
		  "branch_1": {
		  "branch_1_module_1": {
		  "string_arg": "Naame"
		  }
		  }
		}

Если модуль цикла (`whileloop` или `forloop`) состоит из одного внутреннего модуля, результатом модуля цикла будет считаться результат внутреннего модуля.

Сценарии использования переменной

▼ *Пример 1*

В системе запущена обработка 10 инцидентов. В рамках обработки для каждого инцидента выполняется последовательность действий:

1. регистрация инцидента в системе;
2. отправка инцидента по email;
3. удаление инцидента из системы.

Затем нужно проверить результат отправки каждого инцидента по email. Для этого используется значение `results.for_loop_id[0..N].email_send_step_id`.

▼ *Пример 2*

В системе для итерации внутри модуля `forloop` имеется неоднородный список сущностей. Известно, что каждый третий элемент является файлом с определенным атрибутом. Для каждого такого элемента необходимо узнать результат выполнения всей цепочки. Эту информацию можно получить с помощью переменной `results`.

Примеры скриптов и плейбуков

Скрипты

▼ *Скрипт, принимающий и выводящий в консоль any*

```
id: py_dummy_any_arg
name: Py dummy any arg
version: 0.0.1
status: "experimental"
date: ""
author: R-Vision
type: script
source: !py |
  def main(any_arg):
      print(f"We got any_arg: {any_arg}")

      return {"any_arg": any_arg}
```

▼ *Скрипт, принимающий и возвращающий `num` аргумент*

```
id: py_dummy_num_arg
name: Py dummy num arg
version: 0.0.1
status: "experimental"
date: ""
author: R-Vision
type: script
source: !py |
  def main(num_arg: int):
      print(f"We got num_arg: {num_arg}")

      return {"num_arg": num_arg}
```

Плейбук

Плейбук в примере содержит модули:

- модуль `test_forloop` (циклический модуль, делающий одну итерацию). Он состоит из следующих модулей:
  - `while_loop`;

    - скрипт;
    - `all`;
    - `case`.
- модуль `last_module` — скрипт, принимающий и выводящий переменную results полностью.
- модуль `last_module_2` — скрипт, принимающий и выводящий результаты модуля `forloop_m_1` первой итерации модуля `test_forloop`.

▼ *Плейбук*

```
id: playbook_with_forloop_dbg
name: Playbook with forloop dbg
version: 0.0.25
type: playbook
body:
  input:
    properties:
      name:
        type: string
        description: ""
        default: ""
  modules:
    - id: test_forloop
      body:
        type: forloop
        modules:
          -   id: forloop_m_0
              body:
                type: whileloop
                skip_failures: false
                modules:
                  -   id: forloop_m_0_m
                      body:
                          type: script
                          id: py_dummy_num_arg
                          input:
                              num_arg:
                                type: js
                                expr: '3'
                  -   id: forloop_m_1_m
                      body:
                          type: script
                          id: py_dummy_num_arg
                          input:
                              num_arg:
                                type: js
                                expr: '3'
              stop_after_if:
                expr: 3 == 3
                skip_if_stopped: false
```

```
          -   id: forloop_m_1
              body:
                  type: script
                  id: py_dummy_string_arg
                  input:
                      string_arg:
                          type: js
                          expr: playbook_args.name
          -   id: forloop_m_2
              body:
                type: all
                branches:
                  - summary: Branch 1
                    id: branch_1
                    modules:
                      - id: dummy_string_branch_1_module
                        body:
                            type: script
                            id: py_dummy_string_arg
                            input:
                                string_arg:
                                    type: js
                                    expr: playbook_args.name
                parallel: false
                skip_failure: true
          - id: for_loop_m_3
            body:
              type: case
              default:
                -   id: default_case_module
                    body:
                        type: script
                        id: py_dummy_string_arg
                        input:
                            string_arg:
                                type: js
                                expr: playbook_args.name
              branches:
                -   summary: branch_1
                    id: conditional_case_module
                    condition: playbook_args.name == 'Naame'
                    modules:
                      -   id: Dummy module with 1 arg
                          body:
                              type: script
                              id: py_dummy_string_arg
                              input:
                                  string_arg:
                                      type: js
                                      expr: playbook_args.name
        iterator:
```

```
          type: js
          expr: "[2]"
        parallel: false
    -   id: last_module
        body:
            type: script
            id: py_dummy_any_arg
            input:
                any_arg:
                    type: js
                    expr: results
    -   id: last_module_2
        body:
            type: script
            id: py_dummy_any_arg
            input:
                any_arg:
                    type: js
                    expr: results.test_forloop[0].forloop_m_1
```

▼ *Вывод модуля* `last_module`

```
job 862b090a-f513-400f-a52d-04b4151fae0a on worker default_worker_name (tag: all)


--- PYTHON CODE EXECUTION ---

We got any_arg:
{"test_forloop":[{"iter_idx":0,"iter_value":2,"forloop_m_0":[{"iter_idx":0,"forloop_
m_0_m":{"num_arg":3},"forloop_m_1_m":{"num_arg":3}}],"forloop_m_1":{"string_arg":"Na
ame"},"forloop_m_2":{"branch_1":{"dummy_string_branch_1_module":{"string_arg":"Naame
"}}},"for_loop_m_3":{"chosen_branch":"conditional_case_module","conditional_case_mod
ule":{"Dummy module with 1 arg":{"string_arg":"Naame"}}}}]}
```

▼ *Вывод модуля* `last_module_2`

```
job c6ae64ed-3f36-4151-b16c-dd4e14efe447 on worker default_worker_name (tag: all)


--- PYTHON CODE EXECUTION ---

We got any_arg: {"string_arg":"Name"}
```

Схемы конфигурации

В данном разделе представлен обзор структуры и процесса создания схемы конфигурации. Управление схемами конфигурации в системе осуществляется в разделе **Экспертиза**.

О схемах конфигурации

Схемы конфигурации — это структурированные наборы данных, которые используются для задания структуры конфигураций. Конфигурации используются в скриптах и плейбуках. Они необходимы для взаимодействия с внешними системами. Схемы конфигурации во многом сходны с интерфейсами в языках программирования, так как они тоже описывают регламент взаимодействия между компонентами систем.

В схеме содержится детальное описание полей конфигурации — их количество, имена и типы данных. Также схема задает значения полей по умолчанию и определяет, является ли поле обязательным или опциональным.

Схемы конфигурации позволяют решить следующие задачи:

- Предоставление пользователю строго определенного набора полей конфигурации, необходимого для взаимодействия с внешним сервисом или сущностью.
- Валидация данных, введенных пользователем в форме создания конфигурации, в соответствии с заданными свойствами полей.
- Объединение определенных полей конфигурации в одной категории. Это позволяет использовать такие поля совместно.

В настоящий момент схема имеет один уровень вложенности. Это означает, что все поля, перечисленные в схеме, относятся к одному объекту (пространству имен).

При обновлении схемы конфигурации необходимо обновить версию схемы. Все конфигурации, созданные на основе определенной версии схемы, не обновляются при обновлении версии схемы для обеспечения стабильности их работы.

Работа со схемой конфигурации

Доступные операции над схемой конфигурации:

- Создание схемы конфигурации
- Изменение схемы конфигурации
- Удаление схемы конфигурации

Создание схемы конфигурации

Чтобы создать схему конфигурации:

1. Перейдите в раздел **Экспертиза**. Система отобразит список элементов экспертизы.
2. На панели инструментов нажмите на кнопку **Создать** и выберите из выпадающего списка пункт **Схема конфигурации**. Система отобразит окно создания схемы конфигурации.
3. Заполните поля схемы конфигурации, чтобы определить логику ее работы.
4. Нажмите на кнопку **Опубликовать версию**, чтобы сохранить изменения и опубликовать созданную схему конфигурации. Система отобразит уведомление об успешном создании схемы конфигурации. Новая схема конфигурации отобразится в таблице раздела **Экспертиза**.

Чтобы схема конфигурации стала доступной при создании конфигурации, ее необходимо включить.

Вы можете создать схему конфигурации без ее публикации (в виде черновика) с помощью кнопки **Сохранить черновик**. Система отобразит уведомление об успешном создании черновика схемы. Черновик отобразится в таблице раздела **Экспертиза**.

Изменение схемы конфигурации

Чтобы изменить схему конфигурации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку схемы конфигурации в списке. Система отобразит в правой части экрана карточку с подробной информацией о схеме.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Изменить**. Отобразится окно настроек схемы.

Вы также можете открыть окно настроек с помощью кнопки в нижней части карточки:

- Для опубликованной версии: нажмите на кнопку **Просмотр** в нижней части карточки. Система отобразит окно просмотра настроек элемента экспертизы. Нажмите на кнопку **Изменить** в правом нижнем углу окна.
- Для черновика: нажмите на кнопку **Изменить** в нижней части карточки.

4. Внесите требуемые изменения в схему конфигурации.
5. Нажмите на кнопку **Обновить версию**. Новая версия схемы конфигурации будет опубликована. Система отобразит соответствующее уведомление.

При публикации новой версии элемента экспертизы требуется увеличить его текущую версию в поле `version`.

Вы можете сохранить измененную схему конфигурации в виде черновика без публикации помощью кнопки **Сохранить черновик**. Схема будет сохранена для дальнейшего редактирования. В этом случае в верхней части карточки схемы будет отображаться предупреждение о наличии неопубликованных изменений.

Удаление схемы конфигурации

Чтобы удалить схему конфигурации:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Нажмите на строку схемы конфигурации в списке. Система отобразит в правой части экрана карточку с подробной информацией о схеме.
3. Нажмите на кнопку действий (⋮) в верхнем правом углу карточки и выберите опцию **Удалить**.

Отобразится окно подтверждения удаления схемы.

4. Нажмите на кнопку **Удалить**. Система удалит схему и отобразит соответствующее уведомление. Удаленная схема будет исключена из списка элементов экспертизы.

Вы также можете удалить группу элементов экспертизы. Для этого:

1. Перейдите в раздел **Экспертиза**. Система отобразит сведения об имеющихся элементах экспертизы.
2. Установите флажки напротив тех элементов, которые вы хотите удалить, в левом столбце таблицы.
3. Нажмите на кнопку на панели инструментов. Отобразится окно удаления.
4. Нажмите на кнопку **Удалить**. Система удалит выбранные элементы и отобразит соответствующее уведомление. Удаленные элементы будет исключены из списка элементов экспертизы.

Структура схемы конфигурации

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор схемы конфигурации.	строка	да
`name`	Название схемы конфигурации. Допустимы строчные и прописные буквы латинского алфавита (`a–z`, `A–Z`), цифры (`0–9`), точка (`.`), подчеркивание (_) и дефис (`-`) без пробелов.	строка	да
`resource_type`	Тип ресурсов, к которому принадлежит данная схема. Позволяет пользователю различать одинаковые схемы, предназначенные для разных ресурсов. Например, в системе имеются две абсолютно одинаковые и актуальные на данный момент схемы, предназначенные для соединения с разными серверами (Kafka и email). Если в будущем для соединения с сервером Kafka изменится способ входа, принадлежность схем к разным типам позволит отразить это изменение только в схемах, созданных для соединения с сервером Kafka.	строка	да
`version`	Версия схемы конфигурации, представленная в формате Semantic Versioning.	строка	да
`author`	Имя и контактные данные автора схемы конфигурации.	строка	нет

Поле	Описание	Тип данных	Обязательное поле
`description`	Описание схемы конфигурации, содержащее дополнительную информацию о ее назначении и функциональности.	строка	нет
`tags`	Список тегов, присвоенных схеме конфигурации для классификации и быстрого поиска.	массив строк	нет
`schema`	Описания полей схемы конфигурации, используемых для взаимодействия с внешними системами, и их типы данных.	объект	да

Структура описания полей, входящих в поле `schema`

Поле	Описание	Тип данных	Обязательное поле
`name`	Название поля конфигурации, которое позволяет уникально идентифицировать его. Пользователь получает значение этого поля в конфигурации на основе схемы.	строка	да
`type`	Тип данных поля. Допустимые типы данных: `string`, `bool`, `float`, `integer`, `timestamp`, `credentials`, `token`, `connection_string`, `certificate`, `pkcs_container` и `pkcs_12_container`. Ознакомьтесь с разделом Задание типа данных поля конфигурации через переменную.	строка	нет
`required`	Указывает, является ли это поле конфигурации обязательным для заполнения при создании конфигурации. • `true` — поле обязательно для заполнения. • `false` — поле необязательно для заполнения.	логический	да
`default`	Задает значение по умолчанию для этого поля конфигурации в случае, если пользователь не указал его вручную. Если значение поля `default` для поля конфигурации не задано, система вернет `null` в качестве значения поля конфигурации.	строка	нет

Задание типа данных поля конфигурации через переменную

Тип данных поля конфигурации можно задать в формате `$<variable>:<type>`, где `<variable>` — предварительно описанная переменная такого же типа, что и само поле, а `<type>` — тип данных поля. В этом случае поле конфигурации не будет принимать значения, введенные пользователем вручную. Поле принимает только значения в формате ссылок на переменную. Это позволяет конфигурировать определенные поля схемы вне системы.

▼ *Пример*

Для подключения к серверам Kafka создается схема конфигурации, содержащая поле `transport_schema`. В этом поле описывается протокол передачи данных между системой и Kafka.

Фрагмент схемы с описанием поля `transport_schema`:

```
transport_schema
   type: $variable:string
   required: true
```

При использовании протокола HTTP необходимо задать переменную, содержащую значение `http`. Эта переменная будет использоваться во всех конфигурациях для подключения к Kafka, созданных на основе текущей схемы конфигурации.

Если в дальнейшем возникнет необходимость начать использовать протокол HTTPS, а не HTTP, достаточно будет указать новое значение переменной, на которую ссылается тип данных `$variable:string`. Этим новым значением будет `https`. После этого все конфигурации, созданные на основе этой схемы, автоматически начнут использовать для подключения протокол HTTPS.

Схемы расчетов

В данном разделе представлен обзор структуры, принципов работы и процесса создания схем расчетов в системе. Управление схемами расчетов в системе осуществляется в разделе **Экспертиза**.

О схемах расчетов

Схема расчетов — это элемент экспертизы, определяющий формулы, по которым рассчитываются числовые показатели для объектов системы. Например, с помощью схем расчетов можно вычислить комплексный показатель соответствия стандарту на основе оценок требований, входящих в этот стандарт, или среднее время реагирования на инциденты.

Расчеты в схемах поддерживают наследование — возможность создавать расчеты, копирующие свойства наследуемых расчетов за исключением тех, которые явно переопределены.

Результатом расчета по схеме всегда является конечное число.

Чтобы добавить схему расчетов в систему, необходимо выполнить следующие действия:

1. Создать схему расчетов в разделе **Экспертиза**.
2. Определить структуру схемы расчетов.

3. Настроить запуск расчетов по схеме и отображение их результатов в интерфейсе.

Работа со схемой расчетов

Доступные операции над схемой расчетов:

- Создание схемы расчетов
- Изменение схемы расчетов
- Удаление схемы расчетов

Структура схемы расчетов

Схема расчетов содержит обязательные и опциональные поля, определяющие логику расчетов.

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор схемы расчетов, позволяющий ссылаться на нее из других сущностей системы. Должен быть уникальным в пределах экспертизы. Можно использовать UUID в качестве значения.	строка	да
`name`	Название схемы расчетов. Допустимы латинские буквы, цифры, подчеркивания.	строка	да
`description`	Краткое описание назначения схемы расчетов.	строка	нет
`type`	Тип элемента экспертизы. Всегда устанавливается значение `calculation_schema`.	перечисление	да
`tags`	Теги для классификации и поиска схемы в системе.	массив строк	нет
`version`	Версия схемы в формате Semantic Versioning.	строка	да
`status`	Статус текущей версии схемы расчетов. Может принимать любые текстовые значения. Примеры значений: • `debug` — версия в разработке; • `experimental` — экспериментальная версия; • `test` — версия для тестирования; • `stable` — стабильная версия.	строка	нет
`author`	Имя и контактные данные автора схемы.	строка	нет

Поле	Описание	Тип данных	Обязательное поле
`extends`	FQID схемы расчетов, в которой определяются расчеты-наследники для расчетов из текущей схемы.	строка	нет
`calculations`	Массив расчетов. Каждый расчет включает FQID сущностей, к которым применяются расчеты, а также формулы или ссылки на скрипты, которые определяют логику расчетов. См. описание поля.	массив объектов	нет
`summary_calculations`	Массив обобщающих расчетов, в котором задаются и описываются расчеты над несколькими несвязанными друг с другом объектами. См. описание поля.	массив объектов	нет

В схеме расчетов должен быть определен хотя бы один расчет в поле `calculations` или `summary_calculations`.

Расчеты: поле `calculations`

Поле `calculations` содержит объекты расчетов, каждый из которых реализует вычисление числового показателя по заданной формуле. В формуле можно использовать скрипты.

Результатом расчета по схеме всегда является конечное число.

Объект расчета имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор расчета, по которому можно сослаться на расчет. Должен быть уникальным в пределах схемы расчетов. Можно использовать UUID в качестве значения.	строка	да
`name`	Имя расчета, отображаемое в интерфейсе.	строка	да
`description`	Краткое описание назначения расчета.	строка	нет
`entity`	FQID сущности, к которой применяется расчет.	строка	да
`expression`	Математическая формула, по которой выполняется расчет. См. правила описания формулы.	строка	да

Поле	Описание	Тип данных	Обязательное поле
`complexity_limit`	Максимальная сложность расчета. Сложность рассчитывается следующим образом: • базовая сложность выражения в поле `expression` — 1; • каждый скрипт, входящий в выражение, увеличивает его сложность на 10; • к сложности родительского расчета прибавляется сложность всех дочерних расчетов. При превышении предела сложности расчета, указанной в поле `complexity_limit`, расчет завершится с ошибкой. Необязательно точно рассчитывать `complexity_limit` для расчета; можно указать достаточно большое число, например, 100. Если этот предел будет превышен, возникшая ошибка укажет на то, что расчет следует оптимизировать.	положительное целое число	да
`auto_recalculation`	Будет ли расчет автоматически запущен при обновлении свойств или связей объекта, связанного с ним.	булево	да
`children`	Дочерние расчеты.	массив объектов	нет

Правила описания формулы расчета

В поле `expression` следует указать формулу для расчета согласно следующим правилам:

- В формулах поддерживаются все операции, реализованные в библиотеке JavaScript math.js.
- В формулах можно использовать следующие аргументы:
  - целые и дробные числовые константы;
  - все атрибуты сущности, указанной в поле `entity` расчета, имеющие типы `Integer`, `Float`;
  - глобальные переменные типов `Integer`, `Float`;
  - скрипты;
  - расчеты для сущностей, связанных с текущей сущностью. Связь должна быть настроена в

схеме домена в поле `linkages` описания текущей сущности.

Расчеты для связанных сущностей нужно обязательно указать в поле `children`. В таком случае расчет для текущей сущности называется родительским, а расчеты для связанных сущностей — дочерними.

▼ *Пример формулы*

```
- id: test_calc
  name: Тестовый расчет
  entity: test_domain/TestEntity
  expression: "2 * {field:weight} / {variable:divisor}"
  auto_recalculation: false
  complexity_limit: 1
```

Здесь:

- `2` — числовая константа.
- `{field:weight}` — значение поля `weight` сущности `TestEntity`.
- `{variable:divisor}` — значение глобальной переменной `divisor`.

Ссылки на атрибуты, переменные и скрипты следует указывать в фигурных скобках, например, `{variable:divisor}`.

Применение скриптов в расчетах

В поле `expression` можно указать идентификатор скрипта, который будет выполнен при запуске расчета. В расчете будет использоваться значение, возвращенное скриптом.

Результатом работы скрипта должно быть конечное число. Автоматическая валидация возвращаемого значения в системе не предусмотрена, поэтому разработчик скрипта должен реализовать его так, чтобы он возвращал только числа.

▼ *Пример формулы, дополненный вызовом скрипта*

```
- id: test_calc
  name: Тестовый расчет
  entity: test_domain/TestEntity
  expression: |
    mean(
      2 * {field:weight} / {variable:divisor},
      {script:test_calculation_script}
    )
  auto_recalculation: false
  complexity_limit: 11
```

Здесь:

- `{script:test_calculation_script}` — вызов скрипта `test_calculation_script`.

Минимально допустимое значение `complexity_limit` для данного расчета — 11, поскольку к базовой сложности расчета 1 следует прибавить 10 — сложность вызова скрипта.

Дочерние расчеты

Дочерние расчеты — это расчеты для связанных сущностей, участвующие в расчете для текущей сущности (родительском расчете). Ссылки на них следует указать в поле `children` родительского расчета.

Ссылка на дочерний расчет содержит следующие поля:

- `linkage` — идентификатор связи между сущностью, для которой выполняется родительский расчет, и сущностью, для которой выполняется дочерний расчет. Связь должна быть настроена в схеме домена в поле `linkages` описания сущности, для которой выполняется родительский расчет.
- `calculation` — идентификатор дочернего расчета.

Дочерний расчет выполняется для всех сущностей, связанных с текущей, поэтому его результат всегда интерпретируется системой как массив. Чтобы преобразовать результат дочернего расчета в одно число, используйте функции массивов, такие как `min`, `max`, `mean`, `count` или `sum`.

▼ *Пример настройки дочернего расчета*

```
- id: test_calc_with_child
  name: Тестовый расчет с дочерним
  entity: test_domain/TestEntity
  expression: |
    mean(
      2 * {field:weight},
      sum({calculation:child_calc})
    )
  children:
    - linkage: test_domain/TestEntity_ChildEntity
      calculation: child_calc
  auto_recalculation: true
  complexity_limit: 200

- id: child_calc
  name: Дочерний расчет
  entity: test_domain/ChildEntity
  expression: "{script:child_calculation_script}"
  auto_recalculation: true
  complexity_limit: 100
```

Здесь:

- Результат дочернего расчета преобразован в одно число с помощью функции `sum`: `sum({calculation:child_calc})`.
- `test_domain/TestEntity_ChildEntity` — идентификатор связи между сущностями `TestEntity` и `ChildEntity`, заданной в схеме домена `test_domain` в разделе `linkages` описания сущности `TestEntity`.
- `child_calc` — идентификатор дочернего расчета.

Обобщающие расчеты: поле `summary_calculations`

Обобщающие расчеты — это расчеты, позволяющие вычислить метрики для сущностей, для которой нет обобщающей сущности: например, среднее время выполнения задач операторами.

Результатом расчета по схеме всегда является конечное число.

В схеме расчетов обобщающие расчеты настраиваются в поле `summary_calculations`.

В обобщающем расчете можно использовать только результаты других расчетов, созданных в этой же схеме.

Чтобы использовать свойство сущности в обобщающем расчете, это свойство нужно указать в поле `expression` другого расчета.

Обобщающий расчет имеет следующую структуру:

Поле	Описание	Тип данных	Обязательное поле
`id`	Уникальный идентификатор расчета, по которому можно сослаться на расчет. Должен быть уникальным в пределах схемы расчетов. Можно использовать UUID в качестве значения.	строка	да
`name`	Имя расчета, отображаемое в интерфейсе.	строка	да
`description`	Краткое описание назначения расчета.	строка	нет
`expression`	Математическая формула для обобщающего расчета.	строка	да

▼ *Пример обобщающего расчета*

```
calculations:
  - id: task_processing_time
    name: Время обработки задачи
    entity: test_tasks/Task
    expression: "{field:processing_time}"
    auto_recalculation: true
    complexity_limit: 1

summary_calculations:
  - id: mean_task_processing_time
```

```
    name: Среднее время обработки задачи
    expression: "mean({calculation:task_processing_time})"
```

Здесь:

- `task_processing_time` — расчет, возвращающий время выполнения одной задачи.
- `mean_task_processing_time` — обобщающий расчет, возвращающий среднее время выполнения всех задач в системе.

Пример схемы расчетов

В данном разделе приведен пример схемы расчетов.

```
id: evo.compliance.audit.calculations
name: audit_calculations
description: Расчеты для аудитов
type: calculation_schema
tags: [audit, calculations]
version: 1.0.0
status: stable
author: John Doe <johndoe@example.com>
calculations:
  - id: audit_assessment_sum_compliance_index
    name: Сумма индексов соответствия
    entity: evo.compliance.audit/Audit
    expression: |
      round(
        sum({calculation:audit_assessment_compliance_index}),
        {variable:audit_assessment_compliance_index_round}
      )
    children:
      - linkage: evo.compliance.audit/Audits_RequirementValues
        calculation: audit_assessment_compliance_index
    auto_recalculation: true
    complexity_limit: 100

  - id: audit_assessment_compliance_index
    name: Индекс соответствия, вычисленный через скрипт
    entity: evo.compliance.audit/RequirementValues
    expression: "{script:calculate_comp_index}"
    auto_recalculation: true
    complexity_limit: 20

  - id: progress_metric
    name: Прогресс аудита в процентах
    entity: evo.compliance.audit/Audit
    expression: "{field:progress}"
    auto_recalculation: true
    complexity_limit: 1
```

```
summary_calculations:
  - id: mean_audit_progress
    name: Средний прогресс по аудиту
    expression: "mean({calculation:progress_metric})"
```

Интерпретация примера

Метаданные схемы расчетов

- `id`: идентификатор схемы расчетов — `evo.compliance.audit.calculations`.
- `name`: наименование схемы расчетов — `audit_calculations`.
- `description`: описание схемы расчетов — "Расчеты для аудитов".
- `type`: тип элемента экспертизы — `calculation_schema`.
- `tags`: теги схемы расчетов, используемые для классификации — `audit, calculations`.
- `version`: версия схемы расчетов — `1.0.0`.
- `status`: статус разработки схемы — `stable`.
- `author`: автор схемы расчетов — `John Doe <johndoe@example.com>`.

Расчеты

Схема содержит три расчета:

- `audit_assessment_sum_compliance_index` — сумма индексов соответствия требований аудита, округленная до количества разрядов, определенного глобальной переменной `audit_assessment_compliance_index_round`. Индексы соответствия отдельных требований вычисляются с помощью дочернего расчета `audit_assessment_compliance_index`.
- `audit_assessment_compliance_index` — индекс соответствия требования, вычисляемый скриптом `calculate_comp_index`.
- `progress_metric` — расчет, возвращающий значение поля прогресса аудита `progress` для обобщающего расчета `mean_audit_progress`.

Обобщающие расчеты

Схема содержит обобщающий расчет `mean_audit_progress`, вычисляющий средний прогресс по аудитам в системе с помощью расчета `progress_metric`.

Запуск и отображение расчетов по схеме

Расчеты можно запускать:

- при нажатии кнопки, добавленной в интерфейс домена;
- автоматически с помощью триггера;
- при изменении объекта, с которым связан расчет.

Чтобы запустить расчеты по схеме и отобразить их результаты в интерфейсе системы, необходимо

выполнить следующие действия:

1. Если расчеты будут запускаться при нажатии кнопки или по триггеру, следует создать плейбук для запуска расчетов.

Если схема расчетов выключена в разделе **Экспертиза**, при запуске связанного с ней плейбука любым способом расчеты по ней производиться не будут.

2. Настроить метод запуска плейбука расчетов.

Создание плейбука для запуска расчетов

Плейбук для запуска расчетов по схеме должен использовать SDK `calculate`.

▼ *Пример плейбука для запуска расчета*

```
id: calculation_sdk_test_exec
name: Вызов exec через SDK расчётов
version: 1.0.0
author: John Doe <johndoe@example.com>
tags: [audit, calculations]
type: playbook
body:
  input:
    properties: {}
  modules:
    - id: module_0
      body:
        type: rawscript
        input: {}
        source: !bun |
          import * as sdk from '@playbooks/sdk';
          export async function main() {
            // Получение ID текущего объекта из контекста.
            const objectId = await JSON.parse(get_context()).objectId;

            // Формирование объекта с аргументами расчета.
            const command = {
              calculationSchemaId: "calculate_test",
              calculationId: "test",
              object: {
                domainId: "evo.entity.category.standards",
                entityId: "Requirement",
                id: objectId
              },
              cacheFirst: false
            }

            // Выполнение расчета и получение его результатов.
            const result = await sdk.calculation.exec(command);
            console.log('result', result);
```

```
            return result;
          }
```

Настройка метода запуска расчетов

Запуск расчетов при нажатии кнопки

Чтобы запустить расчеты при нажатии кнопки, следует внести изменения в схему домена:

1. создать действие типа `playbook`, в котором указать идентификатор ранее созданного плейбука;
2. создать кнопку (виджет типа `button`) и связать ее с действием;
3. добавить кнопку в представление (например, таблицу или карточку) сущности, для которой производится расчет.

▼ *Пример описания кнопки для запуска расчетов*

```
actions:
  - id: actions.countComplianceIndex
    name: Вычислить индекс соответствия
    type: playbook
    entity: Audit
    playbook: count_compliance_index

views:
  - id: auditView
    type: entity
    entity: Audit

    widgets:
      - id: countComplianceIndexButton
        type: action
        action: actions.countComplianceIndex
        control:
          type: regular_button
          label: "Вычислить индекс соответствия"

    groups:
      - id: infoAuditBlock
        description: Информация об аудите
        type: block
        layout:
          direction: column
          expandable: false
        components:
          - widget: countComplianceIndexButton
```

Интерпретация примера

1. Действие `actions.countComplianceIndex` связано с плейбуком `count_compliance_index`, который

запускает схему расчета индекса соответствия для аудита.

2. Кнопка `countComplianceIndexButton` запускает действие `actions.countComplianceIndex`.
3. Представление карточки аудита `infoAuditBlock` содержит кнопку `countComplianceIndexButton`.

Запуск расчетов с помощью триггера

Можно автоматизировать запуск плейбука расчетов, настроив триггер для него. Поддерживаются оба типа триггеров, доступных в системе — **Расписание** и **Событие триггера**.

Запуск расчетов при изменении связанного объекта

Если в конфигурации расчета установить для свойства `auto_recalculation` значение `true`, при изменении свойств или связей объекта, связанного с расчетом, расчет будет выполнен заново.

Для этого метода запуска создавать плейбук не требуется.

Необходимость перезапуска родительских и дочерних расчетов относительно текущего расчета определяется по следующей логике:

- При перезапуске дочернего расчета родительские расчеты, использующие его результаты, будут выполнены заново.

  Например, расчет для требования — дочерний для расчета по группе требований, а расчет по группе требований — дочерний для расчета по комплексу требований. Тогда при изменении свойств требования будут выполнены заново расчеты для этого требования, а также для группы и комплекса требований, в которые оно входит.

- При перезапуске родительского расчета среди дочерних расчетов, используемых им, будут перезапущены только те, у которых изменились связанные объекты.

  Например, при перезапуске расчета по группе требований будут перезапущены расчеты только для тех требований, свойства которых изменились с прошлого запуска расчета по группе. Результаты остальных расчетов будут получены из системного хранилища результатов расчетов.

Отображение результатов расчетов в интерфейсе

Чтобы отобразить результат расчетов в интерфейсе системы, нужно связать расчет с атрибутом сущности с помощью поля `calculation`, а затем добавить виджет для вывода атрибута в представление сущности.

▼ *Пример вывода результата расчета в карточку требования*

```
entities:
  - id: Requirement
    attributes:
      - id: assessmentIndexCalculation
        name: Результат расчета индекса соответствия
        dataType: Calculation
```

```
        calculation:
          schemaId: evo.compliance.audit.calculations
          id: audit_assessment_compliance_index

...

views:
  - id: requirementView
    type: entity
    entity: Requirement
    widgets:
      - id: editors.assessmentIndexCalculation
        type: attribute
        attribute: assessmentIndexCalculation
        description: Результат расчета индекса соответствия

    groups:
      - id: blocks.info.requirement
        description: Информация о требовании
        type: block
        layout:
          direction: column
          expandable: false
        components:
          - widget: editors.assessmentIndexCalculation
```

Наследование в схемах расчетов

О наследовании в схемах расчетов

В данном разделе описана логика работы наследования расчетов. Используются два основных понятия:

- **Наследуемый расчет** — расчет, на основе которого создается расчет-наследник.
- **Расчет-наследник** — расчет, переопределяющий атрибуты наследуемого расчета и используемый для другой сущности.

Наследование расчетов может быть использовано при работе с сущностями-наследниками.

Например, пусть создана расширенная форма требования аудита, имеющая дополнительные свойства относительно стандартной формы. Тогда можно создать расчет-наследник для индекса соответствия требования, имеющий то же имя, что и расчет для стандартной формы требования, но иную формулу, учитывающую значения дополнительных свойств.

Определять расчеты-наследники можно:

- в той же схеме расчетов, что и наследуемый расчет;

- в другой схеме расчетов.

Определение наследников в текущей схеме расчетов

Чтобы создать расчет-наследник в той же схеме, что и наследуемый расчет, достаточно создать расчет с тем же идентификатором, что и наследуемый, но с другой сущностью в поле `entity`.

▼ *Пример определения наследника в текущей схеме расчетов*

```
id: evo.compliance.audit.calculations
name: audit_calculations
description: Расчеты для аудитов
type: calculation_schema
tags: [audit, calculations]
version: 1.0.0
status: stable
author: John Doe <johndoe@example.com>
calculations:
  - id: audit_assessment_requirement_impact_coefficient
    auto_recalculation: true
    name: Коэффициент значимости требования
    entity: evo.compliance.audit/Requirement
    complexity_limit: 1
    expression: "{field:impactCoefficient}"

  - id: audit_assessment_requirement_impact_coefficient
    auto_recalculation: true
    name: Коэффициент значимости требования
    entity: evo.compliance.audit/RequirementExtendedForm
    complexity_limit: 100
    expression: |
      sum({calculation:audit_assessment_requirement_impact_coefficient_children}) +
      {field:mark}
    children:
      - linkage: evo.compliance.audit/RequirementExtendedForm_Requirement
        calculation: audit_assessment_requirement_impact_coefficient_children

  - id: audit_assessment_requirement_impact_coefficient_children
    auto_recalculation: true
    name: Коэффициент значимости дочернего требования
    entity: evo.compliance.audit/RequirementExtendedForm
    complexity_limit: 1
    expression: "{field:impactCoefficient}"
```

Здесь:

- `audit_assessment_requirement_impact_coefficient` для сущности `Requirement` — наследуемый расчет коэффициента значимости для стандартной сущности требования.
- `audit_assessment_requirement_impact_coefficient` для сущности `RequirementExtendedForm` — расчет-наследник для расширенной сущности требования,

унаследованной от стандартного требования:

- Идентификаторы наследуемого расчета и расчета-наследника совпадают.
- Значения свойств `auto_recalculation` и `name` в расчете-наследнике совпадают с наследуемым расчетом.
- Значения свойств `entity`, `complexity_limit`, `expression` и `children` переопределены.

Определение наследников в другой схеме расчетов

Чтобы создать расчет-наследник в другой схеме расчетов, в метаданных схемы, в которой определяется наследник, необходимо указать ключевое слово `extends` и идентификатор схемы, в которой определен наследуемый расчет.

▼ *Пример определения наследника в другой схеме расчетов*

Схема с наследуемым расчетом

```
id: evo.compliance.audit.calculations
name: audit_calculations
description: Расчеты для аудитов
type: calculation_schema
tags: [audit, calculations]
version: 1.0.0
status: stable
author: John Doe <johndoe@example.com>
calculations:
  - id: audit_assessment_requirement_impact_coefficient
    auto_recalculation: true
    name: Коэффициент значимости требования
    entity: evo.compliance.audit/Requirement
    complexity_limit: 1
    expression: "{field:impactCoefficient}"
```

Здесь:

- `audit_assessment_requirement_impact_coefficient` для сущности `Requirement` — наследуемый расчет коэффициента значимости для стандартной сущности требования.

Схема с расчетом-наследником

```
id: evo.compliance.audit.calculations.extended
name: audit_calculations_extended
description: Расчеты для аудитов (расширенная форма)
type: calculation_schema
tags: [audit, calculations]
version: 1.0.0
status: stable
author: John Doe <johndoe@example.com>
extends: evo.compliance.audit.calculations
calculations:
```

```
  - id: audit_assessment_requirement_impact_coefficient
    auto_recalculation: true
    name: Коэффициент значимости требования
    entity: evo.compliance.audit/RequirementExtendedForm
    complexity_limit: 100
    expression: |
      sum({calculation:audit_assessment_requirement_impact_coefficient_children}) +
      {field:mark}
    children:
      - linkage: evo.compliance.audit/RequirementExtendedForm_Requirement
        calculation: audit_assessment_requirement_impact_coefficient_children

  - id: audit_assessment_requirement_impact_coefficient_children
    auto_recalculation: true
    name: Коэффициент значимости дочернего требования
    entity: evo.compliance.audit/RequirementExtendedForm
    complexity_limit: 1
    expression: "{field:impactCoefficient}"
```

Здесь:

- `extends: evo.compliance.audit.calculations` — ссылка на схему расчетов, содержащую наследуемый расчет.
- `audit_assessment_requirement_impact_coefficient` для сущности `RequirementExtendedForm` — расчет-наследник для расширенной сущности требования, унаследованной от стандартного требования:
  - Идентификаторы наследуемого расчета и расчета-наследника совпадают.
  - Значения свойств `auto_recalculation` и `name` в расчете-наследнике совпадают с наследуемым расчетом.
  - Значения свойств `entity`, `complexity_limit`, `expression` и `children` переопределены.

Тестирование правил экспертизы с помощью языка VRL

В данном разделе рассматриваются основные аспекты создания тестов, включая использование заглушек, проверку полноты тестов, охват всех возможных сценариев и проведение тестов на производительность.

Инструкции по созданию тестов доступны в руководствах по созданию RObject-конфигураций элементов экспертизы.

Заглушки

Заглушки — это специальные заменители, которые используются в тестах для изоляции конкретных участков кода. Они имитируют работу частей системы, которые не поддаются тестированию в данном контексте, например, внешние API или несуществующие компоненты.

В тестах на основе заглушек важно гарантировать, что они правильно настроены, чтобы их использование не привело к ложным срабатываниям и некорректным результатам. Например, если заглушка возвращает неправильные данные, это может исказить результат теста, и его проверка не будет корректной.

Заглушки не должны создавать ложных срабатываний или неправильных выводов в процессе выполнения теста.

Заглушки используются в следующих случаях:

- Изоляция сложных зависимостей: заглушки могут заменить сложные или зависимые компоненты, которые сложно или невозможно протестировать, как, например, внешние веб-сервисы.
- Ускорение тестов: заглушки могут значительно ускорить выполнение тестов, поскольку вместо реальных вызовов к базе данных, файловой системе или сети используется имитация их работы.
- Тестирование ошибок и исключений: заглушки могут быть настроены так, чтобы имитировать ошибки, что полезно для проверки обработки исключительных ситуаций.

Пример 144. Пример теста с заглушкой

```
...
tests:
  - name: Privilege escalation from admin to root
    events:
       - { "uid": "root", "suid": "admin", "cmd": "pytohn", "sproc":"python",
"dproduct": "Ubuntu", "dvchost": "ubnt.land", "dvendor": "Linux", "mode": "0100600",
"msg": [ "audit(1694421703.884:27001658):" ], "name": "", "params": [ "-m" ],
"sourceIp": "10.150.50.115", "suser": "1000", "syscall": "221", "duid": "0"}
    assertion: !vrl |
      true
```

В данном примере заглушка используется для тестирования случая повышения привилегий с пользователя admin до root. В контексте данного теста не происходит выполнения реальных операций с привилегиями или вызовов системы, а проверка сводится к простому утверждению `true`.

Полнота написания тестов

Тесты должны быть достаточно полными, чтобы гарантировать проверку всех ключевых параметров и состояний, особенно при работе с более сложными функциями или обработкой данных.

Ниже приведен пример неполного теста, который проверяет только одно условие (параметр `appname`):

Пример 145. Пример неполного теста

```
...
tests:
  - name: "Test"
```

```
    events:
      - { "raw": {"appname": "testapp", "facility": "kern", "message": "test message"}
}
    assertion: !vrl |
      assert_eq!(.[0].appname, "testapp")
```

Этот тест проверяет только значение поля `appname`, но не включает другие важные параметры, такие как `facility` и `message`. Это может привести к пропуску ошибок в данных.

Оптимизированный тест должен включать проверки для всех важных параметров, таких как `facility` и `message`:

Пример 146. Пример оптимизированного теста

```
...
tests:
  - name: "Test"
    events:
      - { "raw": {"appname": "testapp", "facility": "kern", "message": "test message"}
}
    assertion: !vrl |
      assert_eq!(.[0].appname, "testapp")
      assert_eq!(.[0].facility, "kern")
      assert_eq!(.[0].message, "test message")
```

Рассмотрим пример теста нормализации, где необходимо проверить несколько полей.

Пример 147. Пример неполного теста для проверки нормализации

```
...
normalizer: !vrl |
  .dproduct = "metrics"
  .dvendor = to_string(.raw.namespace) ?? ""
  .name = to_string(.raw.tags.component_id) ?? ""
  .cnt = to_string(.raw.counter.value) ?? ""
  .cat = to_string(.raw.name) ?? ""
tests:
  - name: add_resource1
    events:
     - { "raw":
{"appname":"testtag","facility":"kern","hostname":"host001","message":"13.03.2017
17:54.25 '# UserGate Server# ' 192.168.30.71 -> 188.40.238.250:80
0/440","procid":14884,"severity":"info","timestamp":"2024-02-27T16:14:41Z"}}
    assertion: !vrl |
      assert_eq!(.[0].bytesOut, null)
```

Для того чтобы исправить тест, нужно добавить проверку для каждого поля:

Пример 148. Пример оптимизированного теста

```
...
tests:
  - name: "testing"
    events:
      - raw:
{"counter":{"value":1751.0},"kind":"absolute","name":"component_received_event_bytes_t
otal","namespace":"vector","tags":{"component_id":"unique-id-
12345","component_kind":"transform","component_type":"remap","host":"collector-host-
01"},"timestamp":"2024-05-29T10:26:42.688089484Z"}
    assertion: !vrl |
      assert_eq!(.[0].cat, "component_received_event_bytes_total")
      assert_eq!(.[0].cnt, "1751")
      assert_eq!(.[0].dvendor, "vector")
      assert_eq!(.[0].name, "unique-id-12345")
      assert_eq!(.[0].dproduct, "metrics")
```

Проверка всех сценариев

Тесты должны обеспечивать надежность и предсказуемость правил, чтобы исключить нежелательные срабатывания или пропуски.

Тестирование всех сценариев выполнения правил включает проверку корректной обработки событий, обработки некорректных данных, обработки исключений и проверки работы на граничных условиях.

Тесты можно разделить на несколько категорий:

- **Позитивные сценарии**: проверка, что правило корректно обрабатывает события, соответствующие условиям.
- **Негативные сценарии**: проверка, что некорректные или неподходящие события не приводят к срабатыванию правила.
- **Граничные случаи**: тестирование минимальных, максимальных значений и редких комбинаций.

При описании тестов рекомендуется указывать цель проверки, чтобы упростить сопровождение кода.

Для достижения полной проверки сценариев следует:

- Убедиться, что тесты охватывают все логические ветки правил.
- Проверить реакции правил на отсутствующие, лишние или неверные данные.
- Включить тесты, моделирующие исключительные ситуации, например, пустые события, неожиданные форматы данных.

Пример 149. Пример теста

```
tests:
  - name: "Scenario 1: Basic network connection"
```

```
    description: Проверка корректного определения сетевого подключения.
    events:
      - { "type": "network", "action": "connect", "source_ip": "192.0.2.1" }
    assertion: !vrl |
      assert_eq!(.[0].type, "network") &&
      assert_eq!(.[0].action, "connect") &&
      assert_eq!(.[0].source_ip, "192.0.2.1")

  - name: "Eventid 7: wmic process list using wmiscript.xsl"
    description: Проверка обработки события с использованием `wmic` и `wmiscript.xsl`.
    events:
      - { "dvchost": "host1.domain.local", "fname":
"C:\\Windows\\System32\\wmiscript.dll" }
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].fname, "C:\\Windows\\System32\\wmiscript.dll")

  - name: "Eventid 7: msxsl processing msxscript.xsl"
    description: Проверка корректного распознавания использования `msxsl` и
`msxscript.xsl`.
    events:
      - { "dvchost": "host1.domain.local", "fname":
"C:\\Windows\\System32\\msxscript.dll" }
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].fname, "C:\\Windows\\System32\\msxscript.dll")

  - name: "Eventid 4688: msxsl loading xsl over network"
    description: Проверка загрузки `.xsl` через SMB-сетевой путь.
    events:
      - { "dvchost": "host1.domain.local", "fname": "\\x.x.x.x\\mysmbshare\\file1.xsl"
}
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].fname, "\\x.x.x.x\\mysmbshare\\file1.xsl")

  - name: "Eventid 1: wmic process execution"
    description: Проверка выполнения команды через `wmic`.
    events:
      - { "dvchost": "host1.domain.local", "fname": "C:\\Windows\\System32\\cmd.exe" }
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].fname, "C:\\Windows\\System32\\cmd.exe")

  - name: "Eventid 1: msxsl executing calc.exe"
    description: Проверка выполнения `calc.exe` через `msxsl`.
    events:
      - { "dvchost": "host1.domain.local", "sproc": "C:\\Windows\\System32\\cmd.exe" }
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].sproc, "C:\\Windows\\System32\\cmd.exe")
```

```
  - name: "Eventid 1: msxsl.exe from known hash"
    description: Проверка выполнения `msxsl.exe` из известного пути.
    events:
      - { "dvchost": "host1.domain.local", "dproc": "C:\\System\\Payloads\\msxsl.exe"
}
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].dproc, "C:\\System\\Payloads\\msxsl.exe")

  - name: "Eventid 1: msxsl.exe from random file"
    description: Проверка выполнения произвольного файла вместо ожидаемого.
    events:
      - { "dvchost": "host1.domain.local", "dproc": "C:\\System\\Payloads\\456.exe" }
    assertion: !vrl |
      assert_eq!(.[0].dvchost, "host1.domain.local") &&
      assert_eq!(.[0].dproc, "C:\\System\\Payloads\\456.exe")
```

Тесты на производительность

Производительность правил должна проверяться с помощью специализированных тестов, направленных на выявление узких мест в обработке большого объема событий. Это позволяет гарантировать, что система будет эффективно работать даже при высокой нагрузке и большом объеме данных.

Тесты на производительность обычно включают в себя создание набора событий, который имитирует реальные условия работы системы, и их многократную обработку для оценки времени выполнения правил. Для этого используется параметр `repetitions`, который задает количество повторений для каждого теста, что помогает оценить стабильность работы в условиях высоких нагрузок.

Пример 150. Пример теста на производительность

```
...
benches:
  - name: "Performance Test"
    repetitions: 10000
    events:
      - { "raw": "2022-08-11T12:32:41.000Z 5.86.210.12 - user=user2 act=file op=del
host=pc1 name=pomodoro.txt" }
...
```

В данном примере выполняется тест с 10 000 повторений, где для каждого события генерируется строка, содержащая информацию о файле и операции с ним. Это позволяет проверить, насколько эффективно правило обрабатывает множество таких событий в течение длительного времени.

Справочник по языку запросов RQL

Данный раздел содержит информацию о языке запросов R-Vision Query Language (далее также — RQL). RQL — это декларативный язык запросов, созданный для выполнения операций поиска, анализа и извлечения информации о событиях безопасности. Язык разработан для обработки данных в контексте информационной безопасности и адаптирован для работы с типами данных и моделью данных системы R-Vision SIEM.

Работа с RQL-запросами

В данном разделе описано, как использовать язык запросов RQL для анализа событий и записей активных списков. Описано создание запросов, которые включают работу с проекциями, использование условий, управление выводом, сортировку и группировку результатов, а также заполнение данных. Также описывается использование фильтров и временных ограничений в поиске.

В данной статье при описании выражений приняты следующие обозначения:

- Необязательные компоненты запросов заключается в квадратные скобки.
- `<expression>` — выражение, то есть применение оператора, функция, идентификатор, литерал, арифметическое выражение или выражение в скобках.

Содержание раздела:

- Синтаксис запросов
- Работа с проекциями (оператор `SELECT`)
  - Использование псевдонимов (оператор `AS`)
- Использование условий (оператор `WHERE`)
  - Операторы сравнения
  - Операторы работы с множествами (`IN`) и проверки на пустое значение (`IS NULL`)
    - Поиск по подсетям с оператором `IN`
  - Комбинирование условий (операторы `AND`, `OR`, `NOT`)
  - Работа с настраиваемыми полями универсальной модели
  - Использование фильтров и ограничений по времени
- Группировка (оператор `GROUP BY`)
  - Фильтрация результатов группировки (оператор `HAVING`)
- Сортировка возвращаемых записей (оператор `ORDER BY`)
  - Заполнение пропусков в списке возвращаемых результатов (модификатор WITH FILL)
- Работа с активными списками (операторы `JOIN` и `ENRICH`)
  - Работа с оператором `JOIN` в режиме базового поиска
  - Работа с оператором `JOIN` в режиме продвинутого поиска

    - Работа с оператором ENRICH
- Преобразование значений с помощью условной логики (функция if())
- Максимальное количество и пропуск возвращаемых записей (операторы LIMIT и OFFSET)

Синтаксис запросов

RQL-запросы используются для поиска и извлечения информации о событиях безопасности и записях активных списков.

Работа с событиями в R-Vision осуществляется только в рамках определенного хранилища. Поэтому, в отличие от SQL, в запросах не требуется указывать таблицу базы данных в компоненте FROM <имя таблицы>.

RQL-запросы используются в разделах **Дашборды** и **Поиск**. В разделе **Поиск** доступны два режима поиска — базовый и продвинутый, в последнем случае доступны использование проекций и группировка результатов.

В режиме базового поиска могут опускаться компонент SELECT * и оператор WHERE при указании условия:

```
[SELECT *] [<expression>]
[(INNER | LEFT) JOIN `active_list_name` [AS alias_name] ON <expression> [ENRICH
target_field = source_field]]
[ORDER BY field_name [ASC | DESC] [WITH FILL [FROM <expression> TO <expression>] STEP
<expression>]]
[LIMIT <m> [OFFSET <n>]]
```

```
SELECT <expression> [AS alias_name][, <expression>]
[(INNER | LEFT | RIGHT | FULL) JOIN `active_list_name` [AS alias_name] ON <expression>
[ENRICH target_field = source_field]]
[WHERE <expression>]
[GROUP BY field_name [HAVING <expression>]]
[ORDER BY field_name [ASC | DESC] [WITH FILL [FROM <expression> TO <expression>] STEP
<expression>]]
[LIMIT <m> [OFFSET <n>]]
```

Таблица 14. Соответствие запросов SQL и RQL

Запрос в SQL	Запрос в системе
`SELECT tenantId FROM table`	`SELECT tenantId`
`SELECT tenantId FROM table WHERE length(tenantId) = 5`	`SELECT tenantId WHERE length(tenantId) = 5`

Запрос в SQL	Запрос в системе
`SELECT tenantId, count(*) FROM table GROUP BY tenantId`	`SELECT tenantId, count(*) GROUP BY tenantId`
`SELECT * FROM table WHERE id = 'abcdef'`	`id = 'abcdef'`

Фильтры в разделе **Поиск** и ограничения по времени применяются как дополнительные условия для запроса.

Работа с проекциями (оператор SELECT)

В системе для каждого хранилища событий выбирается модель событий, а для активного списка — схема активного списка, которые определяют структуру данных. По умолчанию система возвращает результат со **всеми** полям текущей модели данных или схемы активного списка независимо от того, какие поля отображаются в списке. Поэтому, если в результат необходимо включить все поля, в базовом поиске компонент запроса `SELECT *` может опускаться.

Оператор `SELECT` используется при работе с проекциями, то есть для указания определенных полей, которые будут включены в результат запроса. Этот оператор часто используется в виджетах и метриках, которые создаются на дашбордах. Имена полей в запросе отделяются друг от друга запятыми.

```
SELECT <expression>[, <expression>]
```

Не ставьте разделитель после последнего поля, включенного в запрос.

Оператор `SELECT` является обязательным при использовании оператора `WHERE` и группировке результатов.

Таблица 15. Соответствие запросов SQL и RQL при использовании проекций

Запрос в SQL	Запрос в системе
`SELECT tenantId, raw FROM table`	`SELECT tenantId, raw`
`SELECT tenantId, raw FROM table WHERE length(tenantId) = 5`	`SELECT tenantId, raw WHERE length(tenantId) = 5`

Использование псевдонимов (оператор AS)

При редактировании виджетов и метрик удобно использовать псевдонимы, которые позволяют сократить названия компонентов запроса в секции **Сопоставление полей**.

```
SELECT <expression> AS alias_name
```

Пример использования псевдонима для функции

При создании виджета типа **Таблица** вводится запрос:

Пример 151. Запрос с использованием псевдонима для функции `count()`

```
SELECT collectorId, count(*) AS cnt WHERE collectorId LIKE '%' GROUP BY collectorId
```

Здесь для функции `count(*)` задается псевдоним `cnt`, который можно указать в секции сопоставления полей.

Использование условий (оператор `WHERE`)

В базовом поиске при задании условия оператор `WHERE` в запросе опускается.

При использовании проекций оператор `WHERE` позволяет применять к указанному полю запрос с выборкой строк по определенному условию.

```
SELECT <expression> WHERE <expression>
```

В условиях используются операторы сравнения, работы с множествами и проверки на пустое значение.

Пример выборки строк по условию

Пример 152. Запрос с оператором `WHERE`

```
SELECT tenantId WHERE length(tenantId) = 5
```

Этот запрос возвращает события с количеством символов в поле `tenantId`, равным 5.

Операторы сравнения

Оператор	Значение
`=`	Равно
`!=`	Неравно
`<`	Меньше
`<=`	Меньше или равно
`>`	Больше
`>=`	Больше или равно
`LIKE`	Соответствует заданному шаблону.
`NOT LIKE`	Не соответствует заданному шаблону.
`BETWEEN`	Входит в диапазон. `a BETWEEN b AND c` равнозначно выражению `a >= b AND a <= c`.

Оператор	Значение
`NOT BETWEEN`	Не входит в диапазон. `a NOT BETWEEN b AND c` равнозначно выражению `a < b OR a > c`.

Шаблон может включать обычные символы и следующие метасимволы:

- `%` — обозначает любое количество произвольных символов или отсутствие символов.
- `_` — обозначает один произвольный символ.
- `\` — используется для экранирования символов `%`, `_` и `\`.

Операторы работы с множествами (`IN`) и проверки на пустое значение (`IS NULL`)

Оператор	Значение
`IN`	Входит во множество
`NOT IN`	Не входит во множество
`IS NULL`	Значение является пустым (NULL)
`IS NOT NULL`	Значение не является пустым (NULL)

Поиск по подсетям с оператором `IN`

Для поиска данных по подсетям используйте оператор `IN`. Формат записи: `%Сетевая маска%/%Префикс%`.

Пример 153. Поиск по подсетям

```
sourceIp IN '::ffff:192.0.2.0/24'
sourceIp IN '2001:db8::/32'
deviceTranslatedAddress IN '203.0.113.0/24'
```

Комбинирование условий (операторы `AND`, `OR`, `NOT`)

Совмещение нескольких условий происходит с использованием операторов:

Оператор	Значение
`AND`	Конъюнкция (И)
`OR`	Дизъюнкция (ИЛИ)
`NOT`	Логическое отрицание

Пример запроса с комбинированным условием

Рассмотрим запрос, возвращающий события, у которых значения в поле `tenantId` начинаются с символа n, а количество символов в поле `raw` превышает 10.

Пример 154. Запрос в SQL

```
SELECT * FROM table WHERE tenantId LIKE 'n%' AND length(raw) > 10
```

Пример 155. Запрос в системе

```
tenantId LIKE 'n%' AND length(raw) > 10
```

Работа с настраиваемыми полями универсальной модели

В RQL-запросах доступно обращение к регулярным настраиваемым полям универсальной модели события. Обращение к таким полям может осуществляться как напрямую, так и посредством специального синтаксиса.

```
<custom_field>:<custom_field_label> <expression>
```

Здесь:

- `<custom_field>` — ключ значения настраиваемого поля, например: `cn1`, `cs1`, `deviceCustomDate1`, `c6a1`.

  Вместо ключа конкретного поля можно использовать его семейство, зависящее от типа данных значения в этом поле:

  - `cs` — `LCString`;
  - `cn` — `UInt64`;
  - `c6a` — `IPv6`;
  - `deviceCustomDate` — `DateTime`.

  Семейство полей рекомендуется использовать в случаях, когда требуется проверять условие `<expression>` по всем полям этого семейства.

- `<custom_field_label>` — поле метаданных, хранящее метку, которая указывает на содержание и назначение связанного с ним поля `<custom_field>`. Эта метка играет роль названия для поля `<custom_field>` и позволяет интерпретировать хранящиеся в нем данные.
- `<custom_field_label>` — поле метаданных, хранящее метку, которая указывает на содержание и назначение связанного с ним поля `<custom_field>`. Эта метка играет роль названия для поля `<custom_field>` и позволяет интерпретировать хранящиеся в нем данные.

  Примеры меток:

  - `cn1Label` — для поля `cn1`;
  - `cs1Label` — для поля `cs1`;
  - `deviceCustomDate1Label` — для поля `deviceCustomDate1`;
  - `c6a1Label` — для поля `c6a1`.

- `<expression>` — выражение с условием поиска по полю значения, использующее операции сравнения (`=`, `!=`, `<>`, `<`, `>`, `<=`, `>=`) и/или операторы `IN`, `LIKE`, `NOT LIKE`.

- Обращение к полю напрямую рекомендуется применять, если достоверно известно, какая информация хранится в этом поле, и требуется получить информацию строго из этого поля.
- Обращение к полям посредством специального синтаксиса рекомендуется применять в случаях, когда оно позволяет упростить RQL-запрос. Например:
  - Когда одно и то же поле универсальной модели содержит различные данные для разных событий.
  - Когда искомые данные распределены по различным полям универсальной модели нескольких событий.

  В этих случаях представленный синтаксис позволяет заменить множественные условия с операторами `AND` и `OR` коротким RQL-запросом.

Примеры обращения к настраиваемым полям

Пример обращения к полю напрямую

Допустим, имеется поток событий, у которых в поле `cs1` хранится адрес электронной почты пользователя, совершившего подозрительное действие. Известно, что часть событий является тестовой и приходит от разных пользователей с одним и тем же электронным адресом `test@example.com`.

Требуется исключить из рассматриваемого потока тестовые события. Ниже представлен вариант соответствующего RQL-запроса с непосредственным обращением к полю `cs1`.

Пример 156. Обращение к настраиваемому полю напрямую

```
cs1 != "test@example.com"
```

Запрос выдаст события, поле `cs1` которых не содержит значение `test@example.com`.

Пример обращения к полю по его ключу

Допустим, имеется поток событий от разных источников. Для событий от некоторых источников в поле `cs1` хранится адрес электронной почты пользователя, совершившего подозрительное действие. В соответствующем поле метки `cs1Label` этих событий хранится значение `email`. Известно, что часть событий от этих источников является тестовой и приходит от разных пользователей с одним и тем же электронным адресом `test@example.com`.

Требуется получить поток событий, для которых указаны электронные адреса пользователей, и исключить из него тестовые события. Ниже представлен RQL-запрос, построенный на основе представленного ранее синтаксиса. Обращение к полю `cs1` выполняется по его ключу.

Пример 157. Обращение к настраиваемому полю по его ключу

```
cs1:email != "test@example.com"
```

Запрос выдаст события, для которых одновременно выполняются следующие условия:

1. Поле `cs1` не содержит значение `test@example.com`.
2. Поле `cs1Label` содержит название `email`.

Представленный запрос эквивалентен следующему запросу:

Пример 158. Обращение к настраиваемому полю и его полю метки

```
cs1 != "test@example.com" AND cs1Label = "email"
```

Пример обращения к полю по его семейству

Допустим, имеется поток событий из разных источников. Для событий некоторых источников в полях семейства `cs` (например, `cs1`, `cs2`, `cs3`) хранится адрес электронной почты пользователя, совершившего подозрительное действие. В соответствующих полях меток (например, `cs1Label`, `cs2Label`, `cs3Label`) этих событий хранится значение `email`. Известно, что часть событий от этих источников является тестовой и приходит от разных пользователей с одним и тем же электронным адресом `test@example.com`.

Требуется получить поток событий, для которых указаны электронные адреса пользователей, и исключить из него тестовые события. Ниже представлен RQL-запрос, построенный на основе представленного ранее синтаксиса. Обращение к настраиваемым полям универсальной модели события выполняется по их семейству `cs`.

Пример 159. Обращение к настраиваемым полям по их семейству

```
cs:email != "test@example.com"
```

Запрос выдаст события, для которых одновременно выполняются следующие условия:

1. Ни одно из полей значений семейства `cs` не содержит значение `test@example.com`.
2. Поле метки, соответствующее полю значений семейства `cs`, содержит название `email`.

Представленный запрос эквивалентен следующему запросу:

Пример 160. Обращение к настраиваемым полям и их полям меток

```
(cs1 != "test@example.com" AND cs1Label = "email") OR (cs2 != "test@example.com" AND
cs2Label = "email") OR (cs3 != "test@example.com" AND cs3Label = "email") OR (cs4 !=
"test@example.com" AND cs4Label = "email") OR (cs5 != "test@example.com" AND cs5Label
= "email") OR (cs6 != "test@example.com" AND cs6Label = "email")
```

Использование фильтров и ограничений по времени

После выполнения запроса в разделе **Поиск** можно отфильтровать возвращенные результаты с помощью кнопки **Добавить фильтр**. В окне добавления фильтра указывается поле, по которому проводится фильтрация, оператор сравнения и значение для сравнения.

Вместе с фильтрами удобно использовать функциональность ограничения запроса по времени. Это ограничение аналогично добавлению к запросу компонента, который ограничивает результаты запроса определенным временным интервалом.

Система воспринимает применяемые фильтры и ограничения по времени как дополнительные условия оператора `WHERE`, присоединяемые оператором `AND`. Переключатель **Инвертировать (NOT)** интерпретируется как оператор `NOT`.

Примеры использования фильтров и ограничения по времени

Пример 1

Вы ввели в строке поиска запрос `tenantId LIKE 'n%' AND length(raw) > 10`. Затем добавили фильтр по полю `originalTimestamp` с оператором `=` и значением `05.05.20236 05:00:00`. Последовательность этих действий равнозначна следующему запросу в строке поиска:

```
tenantId LIKE 'n%' AND length(raw) > 10 AND originalTimestamp = '2023.05.05 05:00:00'
```

Пример 2

Вы хотите ограничить результат запроса `tenantId LIKE 'n%' AND length(raw) > 10` датами `01.01.2023` и `12.12.2023`. Для этого после выполнения запроса выберите значение **Задать период** в поле периода и укажите диапазон периода в полях справа. Эти действия аналогичны следующему запросу:

```
tenantId LIKE 'n%' AND length(raw) > 10 AND (timestamp >= 2023.01.01 AND timestamp <=
2023.12.12)
```

Таким образом, применение фильтров и ограничений по времени существенно упрощает создание поисковых запросов в системе.

Группировка (оператор `GROUP BY`)

Объединять результаты запроса по одному или нескольким полям можно с помощью оператора `GROUP BY`. С оператором `GROUP BY` могут применяться агрегатные функции: `count()`, `sum()`, `avg()`, `max()`, `min()`.

```
SELECT <expression> GROUP BY field_name
```

При использовании оператора GROUP BY необходимо указать, к какой проекции будет применяться объединение. Например, запрос вида `GROUP BY tenantID` воспринимается системой как `SELECT * GROUP BY tenantID`, и поэтому некорректен. Корректный запрос с указанной проекцией имеет вид `SELECT tenantID GROUP BY tenantID`.

Фильтрация результатов группировки (оператор HAVING)

Если требуется отфильтровать группы данных, полученные после объединения результатов, используйте оператор HAVING.

```
SELECT <expression> GROUP BY field_name HAVING <expression>
```

Пример использования оператора HAVING

```
SELECT tenantID, count(*) AS cnt GROUP BY tenantID HAVING cnt > 10
```

Этот запрос выбирает события, группируя их по tenantID, и возвращает только те группы, в которых количество событий больше 10.

Сортировка возвращаемых записей (оператор ORDER BY)

Результаты запроса можно сортировать по любому полю с помощью оператора ORDER BY.

```
[SELECT <expression>] ORDER BY field_name [ASC | DESC]
```

Направление сортировки задается модификаторами:

- ASC — по возрастанию;
- DESC — по убыванию.

Примеры сортировки

Пример 161. Сортировка в базовом поиске

```
ORDER BY collectorId
```

Этот запрос выводит события с сортировкой по полю collectorId в порядке возрастания.

Пример 162. Использование сортировки с агрегатной функцией `count()`

```
SELECT tenantId, count(*) ORDER BY tenantId ASC
```

Этот запрос возвращает события, упорядоченные по возрастанию значения в поле tenantId и указанием количества событий в каждой группе.

Заполнение пропусков в списке возвращаемых результатов (модификатор WITH FILL)

Чтобы заполнить пропуски в списке возвращаемых результатов при сортировке и группировке записей, используйте модификатор WITH FILL. Пропуски заполняются значениями с указанным шагом в указанном промежутке, либо по умолчанию.

`WITH FILL` можно применять только после имени столбца в `ORDER BY`. Допускается использование нескольких `WITH FILL` для разных столбцов.

```
SELECT <expression> ORDER BY field_name WITH FILL [FROM <expression> TO <expression>]
STEP <expression>]]
```

Для случая с несколькими полями `ORDER BY field2 WITH FILL, field1 WITH FILL` порядок заполнения будет соответствовать порядку полей в секции `ORDER BY`.

Здесь:

- `field_name` — имя поля, по которому происходит сортировка.
- `FROM <expression>` — начальное значение интервала, который будет заполнен. Если не указано, будет использовано наименьшее значение в поле `field_name`.
- `TO <expression>` — конечное значение. Если не указано, будет использовано наибольшее значение в поле `field_name`.
- `STEP <expression>` — шаг, с которым будет заполняться интервал. Если не указан, используется значение `1.0` для числовых типов, и 1 секунда для метки времени.

Значение параметра `<expression>` должен принадлежать к одному из следующих типов:

- числовой литерал (целое или дробное число);
- метка времени;
- арифметическая операция с числами, меткой времени, интервалом или функцией;
- функция с возвращаемым значением в виде числа или метки времени.

Примеры заполнения пропусков

Пример 163. Заполнение пропусков для временных интервалов

```
SELECT toStartOfHour(timestamp) as ts, sum(cnt) as sum_cnt GROUP BY ts HAVING sum_cnt
> 0 ORDER BY ts WITH FILL FROM now() - INTERVAL 1 day TO now() STEP INTERVAL 1 hour
```

Данный запрос выполняет следующую последовательность операций:

- `SELECT toStartOfHour(timestamp) as ts, sum(cnt) as sum_cnt`: выбираются значения поля `timestamp`, округленные до начала каждого часа, и вычисляется сумма поля `cnt` для каждой группы записей с одинаковым значением `ts`.
- `GROUP BY ts`: результаты группируются по полю `ts`, которое представляет округленные значения.
- `HAVING sum_cnt > 0`: применяется фильтрация, чтобы оставить только те группы записей, у которых сумма поля `cnt` больше нуля. Это означает, что в результирующем наборе будут только записи, для которых существуют записи с положительными значениями `cnt` внутри группы.
- `ORDER BY ts WITH FILL FROM now() - INTERVAL 1 day TO now() STEP INTERVAL 1 hour`: результаты сортируются по полю `ts`. Здесь используется модификатор `WITH FILL`, который обеспечивает заполнение пропущенных интервалов с шагом в 1 час от текущего времени минус 1 день до

текущего времени. То есть, если в исходных данных отсутствуют записи для определенных интервалов времени, то они будут включены в результирующий набор со значениями по умолчанию.

Пример 164. Заполнение пропусков для временных интервалов с фильтрацией

```
SELECT toStartOfHour(timestamp) as ts, sum(cnt) as sum_cnt GROUP BY ts HAVING sum_cnt
> 0 AND ts >= now() - INTERVAL 1 day AND ts <= now() ORDER BY ts WITH FILL FROM now()
- INTERVAL 1 day TO now() STEP INTERVAL 1 hour
```

Здесь выполняются те же самые операции, что и в первом запросе, но добавляется дополнительный фильтр в выражении `HAVING` и выражении `AND`. Этот фильтр ограничивает результаты только записями, у которых поле `ts` находится в интервале от текущего времени минус 1 день до текущего времени. Таким образом, данный запрос дополнительно фильтрует записи по временному интервалу перед применением сортировки и заполнением пропущенных интервалов.

В результате выполнения этого запроса будет получен отсортированный список, где каждая запись будет содержать округленное значение `timestamp`, сумму `cnt` для соответствующей группы записей, а также пропущенные интервалы времени со значениями по умолчанию, если таковые имеются. В результирующем наборе будут только записи, удовлетворяющие условиям фильтрации по сумме `cnt` и временному интервалу.

Работа с активными списками (операторы `JOIN` и `ENRICH`)

Данные можно обогащать и фильтровать через активные списки с помощью операторов `JOIN` и `ENRICH`:

- В режиме базового поиска для фильтрации записей по данным активных списков используется оператор `JOIN`, который позволяет уточнить результаты запросов на основе совпадения данных без их обогащения. Оператор `ENRICH` используется в комбинации с `JOIN` и позволяет обогащать эти результаты данными из активных списков. Поддерживаются операции `INNER JOIN` и `LEFT JOIN`.
- В режиме продвинутого поиска для объединения данных из активных списков с событиями из хранилища с целью обогащения информации используется оператор `JOIN`. Поддерживаются операции `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN` и `FULL JOIN`.

Работа с оператором `JOIN` в режиме базового поиска

В режиме базового поиска раздела **Поиск** оператор `JOIN` служит для фильтрации записей на основе данных из активных списков. Это позволяет детализировать результаты запросов путем анализа совпадений значений полей без обогащения событий. При этом данные из хранилища событий и активные списки рассматриваются как отдельные таблицы: хранилище событий является "левой таблицей", а активный список — "правой".

```
(INNER | LEFT) JOIN `<active_list_name>` [AS alias_name] ON <expression>
```

Здесь:

- `(INNER | LEFT)` указывает на тип операции `JOIN`:

- `INNER JOIN` отбирает строки при наличии совпадения данных между обеими таблицами.
- `LEFT JOIN` отображает все строки из левой таблицы (хранилище событий), дополняя их совпадающими данными из правой таблицы (активный список), если таковые имеются. Применяется совместно с оператором `ENRICH`.

- `<active_list_name>` обозначает имя активного списка, который используется в качестве правой таблицы для операции `JOIN`.

Имя активного списка необходимо заключать в обратные апострофы.

- `AS alias_name` предоставляет возможность задать псевдоним для активного списка, упрощая дальнейшее обращение к данным в запросе.
- `<expression>` формулирует условие объединения, используя операции сравнения и логические операторы для определения совпадений данных между таблицами.

При обращении к полям таблицы без указания ее названия система автоматически начинает поиск поля в хранилище событий. Если поле в хранилище событий не найдено, поиск поля продолжается в активном списке.

Обращение к полю в записях событий может выполняться одним из следующих способов:

- **Указание только названия поля**. Рекомендуется использовать данный способ, так как поиск полей по умолчанию начинается в "левой" таблице (хранилище событий).
- **Указание имени таблицы из БД ClickHouse и названия поля**. Имя таблицы ClickHouse, связанной с хранилищем событий, необходимо заключать в обратные апострофы, например: `` `clickhouse_table_name`.field ``.

Использование имени хранилища событий вместо имени ассоциированной с ним таблицы ClickHouse не допускается.

Обращение к полю активного списка может выполняться одним из следующих способов:

- **Указание только названия поля**. Данный способ допустим только в тех случаях, когда в записях событий отсутствует поле с таким же названием. В противном случае произойдет обращение к полю в записях событий.
- **Указание имени активного списка и названия поля**. Имя активного списка необходимо заключать в обратные апострофы, например: `` `active_list_name`.field ``.
- **Указание псевдонима активного списка и названия поля**. Псевдоним (alias) активного списка допускается задавать как с обратными апострофами, так и без них, например: `` `alias_name`.field `` или `alias_name.field`.

Пример использования `INNER JOIN` для фильтрации событий

Оператор `INNER JOIN` в разделе **Поиск** может быть использован для фильтрации записей, позволяя ограничить выборку событий на основе соответствия условиям, заданным для данных из активных

списков. Этот механизм не обогащает данные событий новой информацией, но позволяет исключить из результата те события, которые не соответствуют заданным критериям.

Пример запроса ниже демонстрирует использование `INNER JOIN` для фильтрации событий по списку угроз `threat_list`, исключая из результатов все события, не соответствующие IP-адресам, занесенным в этот список. Цель запроса — идентифицировать потенциально вредоносную активность, происходящую с IP-адресов, указанных в списке угроз.

```
INNER JOIN `threat_list` ON sourceIp = `threat_list`.ip_address
```

Этот запрос исключит из результата все события, источник которых не совпадает с IP-адресами, указанными в активном списке `threat_list`. Таким образом, анализ будет сосредоточен исключительно на событиях, источник которых был заранее классифицирован как потенциальная угроза. Поскольку объединение данных не происходит, в результатах запроса отображаются только поля исходных событий, соответствующих условиям фильтрации.

Работа с оператором `JOIN` в режиме продвинутого поиска

В режиме продвинутого поиска раздела **Поиск** оператор `JOIN` позволяет объединять данные из активных списков и хранилища событий, обогащая информацию о событиях. Поддерживаются операции `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN` и `FULL JOIN`, каждая из которых имеет свои особенности при объединении данных. При этом события хранилища и записи активных списков анализируются как данные в двух разных таблицах: хранилище событий служит "левой таблицей", а активный список — "правой".

```
[SELECT <expression>] (INNER | LEFT | RIGHT | FULL) JOIN `<active_list_name>` [AS
alias_name] ON <expression>
```

Здесь:

- `(INNER | LEFT | RIGHT | FULL)` определяет тип операции `JOIN`, применяемой для объединения данных из левой таблицы (хранилища событий) и правой (активного списка):
  - `INNER JOIN` выбирает строки, где присутствует совпадение по заданным условиям в обеих таблицах.
  - `LEFT JOIN` отображает все строки из левой таблицы, дополняя их данными из правой таблицы при наличии совпадений.
  - `RIGHT JOIN` аналогичен `LEFT JOIN`, но выводит все строки из правой таблицы, дополняя их данными из левой таблицы при наличии совпадений.
  - `FULL JOIN` объединяет методы `LEFT JOIN` и `RIGHT JOIN`, показывая все строки из обеих таблиц.
- `<active_list_name>` обозначает имя активного списка, который используется в качестве правой таблицы.

  Имя активного списка необходимо заключать в обратные апострофы.

- `[AS alias_name]` позволяет назначить псевдоним активному списку, упрощая дальнейшую работу с запросом.

- `<expression>` обозначает условие для объединения, которое может включать операции сравнения и логические операторы, для определения критериев совпадения данных между таблицами.

При обращении к полям таблицы без указания ее названия система автоматически начинает поиск поля в хранилище событий. Если поле в хранилище событий не найдено, поиск поля продолжается в активном списке. Обращение к полю в записях событий может выполняться одним из следующих способов:

- **Указание только названия поля**. Рекомендуется использовать данный способ, так как поиск полей по умолчанию начинается в "левой" таблице (хранилище событий).
- **Указание имени таблицы из БД ClickHouse и названия поля**. Имя таблицы ClickHouse, связанной с хранилищем событий, необходимо заключать в обратные апострофы, например: `` `clickhouse_table_name`.field ``.

  Использование имени хранилища событий вместо имени ассоциированной с ним таблицы ClickHouse не допускается.

Обращение к полю активного списка может выполняться одним из следующих способов:

- **Указание только названия поля**. Данный способ допустим только в тех случаях, когда в записях событий отсутствует поле с таким же названием. В противном случае произойдет обращение к полю в записях событий.
- **Указание имени активного списка и названия поля**. Имя активного списка необходимо заключать в обратные апострофы, например: `` `active_list_name`.field ``.
- **Указание псевдонима активного списка и названия поля**. Псевдоним (alias) активного списка допускается задавать как с обратными апострофами, так и без них, например: `` `alias_name`.field `` или `alias_name.field`.

Примеры использования `JOIN` в продвинутом поиске

Пример 1. Использование `INNER JOIN`

Оператор `INNER JOIN` может быть использован для объединения данных о событиях с активными списками, что позволит идентифицировать события, исходящие с IP-адресов, внесенных в список блокировки.

Пример запроса ниже демонстрирует использование `INNER JOIN` для соединения данных событий с активным списком `blacklist_ips` по IP-адресу источника (`sourceIp`), с целью выявления событий с блокированными IP-адресами.

```
SELECT sourceIp, timestamp, action, ip_address
INNER JOIN `blacklist_ips` ON sourceIp = `blacklist_ips`.ip_address
```

Визуализация ожидаемого результата запроса:

event.sourceIp	event.timestamp	event.action	blacklist_ips.ip_address
192.0.2.1	2022-07-01T12:00:00Z	Blocked	192.0.2.1
192.0.2.2	2022-07-02T12:00:00Z	Blocked	192.0.2.2

В этой таблице отображаются события, произошедшие с IP-адресов, указанных в системе (`event`), и соответствующие IP-адресам в активном списке `blacklist_ips`. Используя `INNER JOIN`, выборка ограничивается событиями, источники которых находятся в списке блокировки.

Пример 2. Использование `LEFT JOIN`

Оператор `LEFT JOIN` может быть использован для объединения данных о событиях входа в систему с информацией о пользователях из активного списка. Это позволяет получить расширенный контекст активности пользователей, включая отсутствие событий для определенных пользователей. В случаях, когда соответствующая запись в активном списке отсутствует, результатом объединения для данной записи события будут значения `NULL` в полях, предназначенных для данных из активного списка.

Пример запроса ниже демонстрирует использование `LEFT JOIN` для соединения событий с записями активного списка о пользователях `user_info` по идентификатору пользователя. Цель — включить в анализ всех пользователей, включая тех, по которым отсутствуют данные о событиях входа.

```
SELECT sourceIp, timestamp, ip_address, userName
LEFT JOIN `user_info` ON userId = `user_info`.user_id
```

Визуализация ожидаемого результата запроса:

event.sourceIp	event.timestamp	user_info.ip_address	user_info.userName
192.0.2.1	2022-07-01T12:00:00Z	192.0.2.1	Sergey
192.0.2.2	2022-07-02T12:00:00Z	192.0.2.2	John
192.0.2.3	2022-07-03T12:00:00Z	NULL	NULL

В этой таблице показаны все попытки входа пользователей, зафиксированные в системе (`event`), дополненные информацией о них из активного списка `user_info`. Последняя строка демонстрирует случай, когда в таблице событий имеются данные о входе с IP-адреса 192.0.2.3, но информация о пользователе, выполнившем вход, отсутствует в активном списке `user_info`.

Пример 3. Использование `RIGHT JOIN`

Оператор `RIGHT JOIN` может быть применен для объединения информации об устройствах из активного списка с данными о событиях, позволяя анализировать данные обо всех устройствах независимо от наличия связанных с ними событий. В случаях, когда соответствующая запись в хранилище отсутствует, результатом объединения для данной записи события будут значения `NULL` в полях, предназначенных для данных из хранилища.

Пример запроса ниже демонстрирует использование `RIGHT JOIN` для соединения информации об

устройствах из активного списка `device_info` с данными о событиях на основе совпадения серийных номеров устройств (`serial_number`) с внешними идентификаторами устройств в событиях (`deviceExternalId`). Цель — предоставить полный обзор по всем устройствам, включая те, по которым отсутствуют события.

```
RIGHT JOIN `device_info` ON deviceExternalId = `device_info`.serial_number
```

Визуализация ожидаемого результата запроса:

event.deviceExternalId	event.action	device_info.serial_number	device_info.device_name
12345	Alert	12345	DeviceA
NULL	NULL	67890	DeviceB

В этой таблице отображается информация об устройствах из активного списка `device_info`, дополненная данными о событиях из системы. Запись устройства `DeviceB` демонстрирует случай, когда для устройства нет зарегистрированных событий.

Пример 4. Использование `FULL JOIN`

Оператор `FULL JOIN` можно использовать для комплексного анализа данных, чтобы объединить информацию о событиях с данными из активных списков, не исключая записи без взаимных соответствий. Это обеспечивает возможность анализа, включая изучение как связанных, так и независимых данных событий и элементов активных списков.

Пример запроса ниже иллюстрирует использование `FULL JOIN` для объединения данных о событиях с информацией из активного списка `network_segments` на основе соответствия между IP-адресами устройств (`deviceIp`) и сегментами сети (`segment_ip`). Цель — предоставить полный обзор сетевой активности и конфигурации сети, включая события и сетевые сегменты, которые прямо не связаны.

```
FULL JOIN `network_segments` ON deviceIp = `network_segments`.segment_ip
```

Визуализация ожидаемого результата запроса:

event.deviceIp	event.action	network_segments.segment_ip	network_segments.segment_name
198.51.100.1	Access_Granted	198.51.100.1	Office_Network
NULL	NULL	198.51.100.1	Data_Center
203.0.113.24	Unauthorized_Access	NULL	NULL

В этой таблице представлена информация, объединяющая события сетевой активности и конфигурацию сетевых сегментов. Запись для сегмента `Data_Center` показывает сетевой сегмент без зарегистрированных событий, тогда как запись для IP-адреса `203.0.113.24` демонстрирует событие, для которого нет соответствующего сегмента сети в активном списке.

Работа с оператором ENRICH

Оператор ENRICH дает возможность заменять или добавлять значения в полях событий на основе данных из соответствующих записей активного списка. Он используется в сочетании с оператором JOIN. ENRICH применяется только после того, как условие соединения, заданное в JOIN, будет выполнено.

```
[SELECT <expression>]
(INNER | LEFT | RIGHT | FULL) JOIN `<active_list_name>` ON <join_condition>
ENRICH <target_field> = <source_field>
```

Здесь:

- <active_list_name> — имя активного списка, данные из которого используются для обогащения.

  Имя активного списка необходимо заключать в обратные апострофы.

- <join_condition> — условие соединения, определяющее правила поиска совпадений между данными событий и записями в активном списке.

  При обращении к полям таблицы без указания ее названия система автоматически начинает поиск поля в хранилище событий. Если поле в хранилище событий не найдено, поиск поля продолжается в активном списке. Обращение к полю в записях событий может выполняться одним из следующих способов:

  - **Указание только названия поля**. Рекомендуется использовать данный способ, так как поиск полей по умолчанию начинается в "левой" таблице (хранилище событий).
  - **Указание имени таблицы из БД ClickHouse и названия поля**. Имя таблицы ClickHouse, связанной с хранилищем событий, необходимо заключать в обратные апострофы, например: `clickhouse_table_name`.field.

  Использование имени хранилища событий вместо имени ассоциированной с ним таблицы ClickHouse не допускается.

  Обращение к полю активного списка может выполняться одним из следующих способов:

  - **Указание только названия поля**. Данный способ допустим только в тех случаях, когда в записях событий отсутствует поле с таким же названием. В противном случае произойдет обращение к полю в записях событий.
  - **Указание имени активного списка и названия поля**. Имя активного списка необходимо заключать в обратные апострофы, например: `active_list_name`.field.
  - **Указание псевдонима активного списка и названия поля**. Псевдоним (alias) активного списка допускается задавать как с обратными апострофами, так и без них, например: `alias_name`.field или alias_name.field.

- `<target_field>` — поле в записях событий, для которого будет выполнено обогащение.

Обращение к полю в записях событий может выполняться одним из следующих способов:

- **Указание только названия поля**. Рекомендуется использовать данный способ, так как поиск полей по умолчанию начинается в "левой" таблице (хранилище событий).
- **Указание имени таблицы из БД ClickHouse и названия поля**. Имя таблицы ClickHouse, связанной с хранилищем событий, необходимо заключать в обратные апострофы, например: `` `clickhouse_table_name`.field ``.

  Использование имени хранилища событий вместо имени ассоциированной с ним таблицы ClickHouse не допускается.

- `<source_field>` — поле в активном списке, значения из которого используются для обогащения данных событий.

Обращение к полю активного списка может выполняться одним из следующих способов:

- **Указание только названия поля**. Данный способ допустим только в тех случаях, когда в записях событий отсутствует поле с таким же названием. В противном случае произойдет обращение к полю в записях событий.
- **Указание имени активного списка и названия поля**. Имя активного списка необходимо заключать в обратные апострофы, например: `` `active_list_name`.field ``.
- **Указание псевдонима активного списка и названия поля**. Псевдоним (alias) активного списка допускается задавать как с обратными апострофами, так и без них, например: `` `alias_name`.field `` или `alias_name.field`.

Оператор `ENRICH` позволяет обогащать данные событий из нескольких активных списков в рамках одного запроса. Для точного указания полей из конкретных активных списков следует использовать полное имя поля, включающее имя активного списка и имя поля, разделенные точкой.

Примеры использования `ENRICH`

Обогащение данных события значениями из одного активного списка

Пример 165. Использование `ENRICH`

```
LEFT JOIN `user_info` ON userId = `user_info`.name
ENRICH id = `user_info`.age
```

В данном примере запрос обогащает значение поля `id` в записях событий значением поля `age` из активного списка `user_info`, если значение поля `userId` события совпадает с полем `name` в активном списке.

Обогащение данных события дополнительными данными из активных списков

Применение запросов с `INNER JOIN` и `ENRICH` в разделе **Поиск** позволяет обогатить данные событий дополнительными данными из активных списков при условии точного совпадения между записями событий и данными активного списка. В отличие от `LEFT JOIN`, `INNER JOIN` выбирает только те записи, для которых существуют совпадающие данные в обеих таблицах, и далее `ENRICH` используется для обогащения этих данных.

Этот метод подходит для сценариев, где необходимо ограничить анализ событиями, которые имеют прямое соответствие в активных списках, и одновременно обогатить эти события дополнительной информацией из этих списков.

Пример 166. Использование `INNER JOIN` с `ENRICH`

```
INNER JOIN `device_info` ON deviceId = `device_info`.device_id
ENRICH deviceType = `device_info`.type
```

Обращение к нескольким активным спискам

Пример 167. Использование `LEFT JOIN` с `ENRICH`

```
SELECT id
LEFT JOIN `admin_list` ON userId = `admin_list`.user_id
LEFT JOIN `security_events` ON eventCode = `security_events`.code
ENRICH adminName = `admin_list`.name, securityLevel = `security_events`.level
```

В данном примере запрос обогащает данные событий, используя информацию из двух разных активных списков — `admin_list` и `security_events`. Оператор `ENRICH` применяется для замены (или добавления, если такого поля нет) в записях событий: значения поля `adminName` на основании данных об имени администратора из списка `admin_list` и значения поля `securityLevel` на основании уровня безопасности из списка `security_events`. Оба обогащения основываются на совпадении условий: `userId` с `user_id` для списка администраторов и `eventCode` с `code` для событий.

Преобразование значений с помощью условной логики (функция `if()`)

Функция `if()` позволяет отображать в результатах запроса преобразованные значения вместо исходных. Это удобно для категоризации результатов или более наглядного представления данных на дашбордах и в отчетах.

```
SELECT if(cond, then, else) AS field1,
...
[GROUP BY field1
ORDER BY field1]
```

Здесь:

- `cond` — проверяемое условие;
- `then` — выражение, результат которого функция вернет, если `cond` истинно;
- `else` — выражение, результат которого возвращается, если `cond` ложно.

В качестве `else` можно использовать вложенную функцию `if()`. Для компактного представления нескольких условий можно воспользоваться функцией `multiIf()`:

```
multiIf(cond_1, then_1, cond_2, then_2, ... else)
```

Здесь:

- `cond_N` — условие.
- `then_N` — результат функции при выполнении условия `cond_N`.
- `else` — результат функции, если ни одно из условий не выполнено.

Пример использования условных функций

Рассмотрим запрос, который возвращает данные, сгруппированные по нескольким полям, вместе с количеством записей:

```
SELECT
    ipAddress,
    deviceName,
    status,
    count(*) AS total_checks
GROUP BY ipAddress, deviceName, status
ORDER BY ipAddress, deviceName, status;
```

Визуализация ожидаемого результата запроса:

ipAddress	deviceName	status	total_checks
192.0.2.0	DeviceA	PASSED	597
192.0.2.0	DeviceA	NOT_PASSED	543
192.0.2.0	DeviceA	NOT_COMPLETED	1116
192.0.2.1	DeviceB	PASSED	56
192.0.2.1	DeviceB	NOT_PASSED	68
192.0.2.1	DeviceB	NOT_COMPLETED	147

Чтобы отобразить в столбце `status` значения на русском языке, преобразуем запрос следующим образом:

```
SELECT
  ipAddress,
  deviceName,
  if(
      equals(status, "PASSED"),
      "Пройденные",
      if(
          equals(status, "NOT_PASSED"),
```

```
          "Непройденные",
          if(
              equals(status, "NOT_COMPLETED"),
              "Неприменимые результаты",
              "-"
              )
        )
    ) AS status,
count(*) AS total_checks
GROUP BY ipAddress, deviceName, status
ORDER BY ipAddress, deviceName, status;
```

Здесь равенство поля `status` и указанного значения проверяется с помощью функции `equals()`. Если `equals()` возвращает ложное значение, происходит переход ко вложенному `if()`.

Визуализация ожидаемого результата запроса:

ipAddress	deviceName	status	total_checks
192.0.2.0	DeviceA	Пройденные	597
192.0.2.0	DeviceA	Непройденные	543
192.0.2.0	DeviceA	Неприменимые результаты	1116
192.0.2.1	DeviceB	Пройденные	56
192.0.2.1	DeviceB	Непройденные	68
192.0.2.1	DeviceB	Неприменимые результаты	147

Чтобы упростить запись, используем функцию `multiIf()`:

```
SELECT
  ipAddress,
  deviceName,
  multiIf(
      equals(status, "PASSED"), "Пройденные",
      equals(status, "NOT_PASSED"), "Непройденные",
      equals(status, "NOT_COMPLETED"), "Неприменимые результаты",
      "-"
      ) AS status,
count(*) AS total_checks
GROUP BY ipAddress, deviceName, status
ORDER BY ipAddress, deviceName, status;
```

Максимальное количество и пропуск возвращаемых записей (операторы `LIMIT` и `OFFSET`)

По умолчанию в списке результатов поискового запроса отображается 500 записей. Однако вы

можете ограничить количество возвращаемых записей с помощью оператора `LIMIT`.

```
LIMIT <m> [OFFSET <n>]
```

Например, чтобы ограничить выдачу двадцатью записями, используется запрос:

```
LIMIT 20
```

Также можно добавить команду `OFFSET`, которая указывает, сколько строк необходимо пропустить перед началом вывода. Это полезно, если вы хотите начать вывод данных с определенной строки. Например, чтобы пропустить первые 10 строк и вывести следующие 20, используется запрос:

```
LIMIT 20 OFFSET 10
```

Функции

В RQL функции делятся на два основных типа: обычные функции и агрегатные функции, которые обладают различным поведением и областью применения.

Обычные функции применяются к каждой строке индивидуально, вычисляя результат для каждой строки без учета других строк данных. В отличие от них, агрегатные функции обрабатывают данные сразу по нескольким строкам, аккумулируя значения и вычисляя общий результат для набора строк.

Типизация в RQL является строгой, что означает отсутствие неявных преобразований между типами данных. Все функции определены для конкретных типов аргументов, следовательно, иногда может потребоваться использование функций преобразования типов для корректной работы.

В RQL выражения с одинаковым абстрактным синтаксическим деревом (AST) в рамках одного запроса исполняются однократно. Это включает в себя одинаковые подзапросы, которые также оптимизируются путем их объединения и однократного выполнения.

Функции в RQL возвращают строго одно значение, тип которого определяется типами аргументов, но не их конкретными значениями. Исключением являются функция toFixedString.

При работе с функциями важно учитывать обработку значений `NULL`. Если хотя бы один из аргументов функции является `NULL`, то результатом функции, как правило, также будет `NULL`, если иное не указано в описании функции.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций для версии ClickHouse 24.7.

Агрегатные функции

Агрегатные функции в RQL позволяют проводить расчеты над множествами значений, предоставляя доступ к вычислению общих сумм, средних, максимальных и минимальных значений, а также других агрегатов из больших объемов данных. Их ключевая особенность — это возможность эффективно обработать информацию, полученную из различных строк данных.

Обработка `NULL` в агрегатных функциях предполагает исключение `NULL` значений из обработки, то есть, все `NULL` значения пропускаются и не участвуют в агрегации.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список агрегатных функций для ClickHouse 24.7.

Таблица 16. Агрегатные функции

Функция	Описание
`count()`	Подсчитывает количество строк, удовлетворяющих условиям запроса (`count()`). Если условие не указано, подсчитывает все строки в наборе данных.
`min()`	Возвращает минимальное значение в столбце (`min(column)`). Может применяться к числовым, строковым и некоторым другим типам данных.
`max()`	Возвращает максимальное значение в столбце (`max(column)`). Поддерживает такие же типы данных, как и функция `min()`.
`sum()`	Суммирует значения по столбцу (`sum(column)`). Применима только к числовым типам данных.
`avg()`	Вычисляет среднее значение для столбца, содержащего числовые данные (`avg(column)`).
`any()`	Возвращает произвольное значение из столбца (`any(column)`). Может быть использована для оптимизации, когда точное значение не важно.
`stddevPop()`	Вычисляет стандартное отклонение для всей выборки (`stddevPop(column)`). Поддерживается только для числовых данных.
`stddevSamp()`	Вычисляет выборочное стандартное отклонение (`stddevSamp(column)`). Используется для оценки стандартного отклонения на основе выборки из генеральной совокупности.
`varPop()`	Вычисляет дисперсию для всей выборки (`varPop(column)`).
`varSamp()`	Вычисляет дисперсию выборки (`varSamp(column)`), что полезно для анализа вариативности данных в выборке.
`corr()`	Вычисляет коэффициент корреляции Пирсона между двумя столбцами (`corr(column1, column2)`).

Функция	Описание
`covarPop()`	Вычисляет ковариацию для всей совокупности (`covarPop(column1, column2)`).
`covarSamp()`	Вычисляет ковариацию на основе выборки из генеральной совокупности (`covarSamp(column1, column2)`).
`entropy()`	Вычисляет энтропию для столбца (`entropy(column)`), что может использоваться для оценки степени разнообразия или неопределенности в данных.
`intervalLengthSum()`	Суммирует длины интервалов между значениями в столбце (`intervalLengthSum(startColumn, endColumn)`), что может применяться для вычисления общего времени или расстояния.
`median()`	Вычисляет медиану значений в столбце (`median(column)`).
`rankCorr()`	Вычисляет ранговую корреляцию между двумя переменными (`rankCorr(column1, column2)`).
`sumKahan()`	Суммирует числа с использованием алгоритма Кахана для уменьшения ошибки округления (`sumKahan(column)`).
`studentTTest()`	Выполняет t-тест Стьюдента для сравнения средних двух выборок. Используется без параметра `confidenceLevel` (`studentTTest(sample1, sample2)`).
`welchTTest()`	Выполняет t-тест Уэлча, модификацию t-теста Стьюдента, предназначенную для использования с выборками разного размера и дисперсии. Используется без параметра `confidenceLevel` (`welchTTest(sample1, sample2)`).
`anyHeavy()`	Возвращает наиболее часто встречающееся значение в столбце (`anyHeavy(column)`).
`anyLast()`	Возвращает последнее значение в столбце (`anyLast(column)`).
`boundingRatio()`	Вычисляет отношение ограничивающего объема (`boundingRatio(column)`), что может использоваться для оценки плотности данных.
`first_value()`	Возвращает первое значение в порядке сортировки (`first_value(column)`).
`last_value()`	Возвращает последнее значение в порядке сортировки (`last_value(column)`).
`argMin()`	Возвращает значение аргумента, при котором достигается минимальное значение выражения (`argMin(arg, val)`).
`argMax()`	Возвращает значение аргумента, при котором достигается максимальное значение выражения (`argMax(arg, val)`).
`avgWeighted()`	Вычисляет взвешенное среднее значение (`avgWeighted(value, weight)`).

Функция	Описание
`deltaSum()`	Вычисляет сумму разностей последовательных значений в столбце (`deltaSum(column)`), полезно для анализа изменений значений во времени.
`deltaSumTimestamp()`	Вычисляет сумму разностей между последовательными временными метками (`deltaSumTimestamp(timestampColumn)`), используется для вычисления общего времени между событиями.
`groupArray()`	Агрегирует значения в массив (`groupArray(column)`), обеспечивает возможность работы с массивом значений как с единой сущностью.
`groupUniqArray()`	Агрегирует уникальные значения в массив (`groupUniqArray(column)`), используется для удаления дубликатов из набора данных.
`groupArrayInsertAt()`	Вставляет значение в массив по указанному индексу (`groupArrayInsertAt(value, index)`), позволяет модифицировать массивы при агрегации.
`groupArraySorted()`	Сортирует элементы массива (`groupArraySorted(column)`), обеспечивает порядок данных в массиве.
`groupArrayIntersect()`	Возвращает массив, содержащий только элементы, присутствующие во всех входных массивах (`groupArrayIntersect(arrays)`), полезно для поиска общих элементов.
`groupBitAnd()`	Выполняет побитовую операцию И над всеми значениями столбца (`groupBitAnd(column)`), применяется для агрегации битовых флагов.
`groupBitOr()`	Выполняет побитовую операцию ИЛИ над всеми значениями столбца (`groupBitOr(column)`), используется для объединения битовых флагов.
`groupBitXor()`	Выполняет побитовую операцию исключающее ИЛИ над всеми значениями столбца (`groupBitXor(column)`), полезно для вычисления различий между битовыми флагами.
`groupBitmap()`	Агрегирует значения в битовую карту (`groupBitmap(column)`), эффективно для подсчета уникальных значений.
`groupBitmapAnd()`	Выполняет операцию И над битовыми картами (`groupBitmapAnd(bitmap1, bitmap2)`), используется для поиска общих уникальных значений между наборами.
`groupBitmapOr()`	Выполняет операцию ИЛИ над битовыми картами (`groupBitmapOr(bitmap1, bitmap2)`), применяется для объединения уникальных значений из нескольких наборов.
`groupBitmapXor()`	Выполняет операцию исключающее ИЛИ над битовыми картами (`groupBitmapXor(bitmap1, bitmap2)`), полезно для определения уникальных значений, отличающихся между наборами.
`sumWithOverflow()`	Суммирует значения, позволяя переполнение (`sumWithOverflow(column)`), применяется для обработки очень больших сумм, где может произойти переполнение.

Функция	Описание
`sumMap()`	Суммирует элементы входных массивов, группируя по ключам (`sumMap(keysArray, valuesArray)`), полезно для агрегации данных по ключу.
`minMap()`	Находит минимальные элементы входных массивов, группируя по ключам (`minMap(keysArray, valuesArray)`), используется для определения минимального значения по ключу.
`maxMap()`	Находит максимальные элементы входных массивов, группируя по ключам (`maxMap(keysArray, valuesArray)`), применяется для определения максимального значения по ключу.
`skewSamp()`	Вычисляет выборочный коэффициент асимметрии для набора данных (`skewSamp(column)`), используется для оценки асимметрии распределения.
`skewPop()`	Вычисляет коэффициент асимметрии для всей совокупности (`skewPop(column)`), полезно для анализа формы распределения на всей выборке.
`kurtSamp()`	Вычисляет выборочный коэффициент эксцесса (`kurtSamp(column)`), используется для анализа "остроты" пика распределения данных.
`kurtPop()`	Вычисляет коэффициент эксцесса для всей выборки (`kurtPop(column)`), позволяет оценить, насколько плоское или острое распределение у всей совокупности.
`uniq()`	Подсчитывает количество уникальных значений в столбце (`uniq(column)`), используется для определения количества различных элементов.
`uniqExact()`	Точно подсчитывает количество уникальных значений (`uniqExact(column)`), предпочтительно для небольших наборов данных, где требуется точность.
`uniqCombined()`	Приблизительно подсчитывает количество уникальных значений с помощью алгоритма HyperLogLog (`uniqCombined(column)`), эффективно для больших объемов данных.
`uniqCombined64()`	Приблизительно подсчитывает количество уникальных значений с помощью 64-битной версии алгоритма HyperLogLog (`uniqCombined64(column)`), предназначено для очень больших наборов данных.
`uniqHLL12()`	Приблизительно подсчитывает количество уникальных значений с использованием алгоритма HyperLogLog с точностью 12 бит (`uniqHLL12(column)`), используется для оптимизации расчета уникальности в больших наборах данных.
`uniqTheta()`	Приблизительно подсчитывает количество уникальных значений с использованием алгоритма Theta Sketch (`uniqTheta(column)`), предлагает улучшенную производительность и точность на больших объемах данных.

Функция	Описание
`simpleLinearRegression()`	Вычисляет параметры простой линейной регрессии для двух переменных (`simpleLinearRegression(y, x)`), позволяя оценить линейную зависимость между ними.
`stochasticLinearRegression()`	Вычисляет параметры стохастической линейной регрессии (`stochasticLinearRegression(y, x)`), используется для моделирования линейной зависимости с учетом случайного компонента.
`stochasticLogisticRegression()`	Вычисляет параметры стохастической логистической регрессии (`stochasticLogisticRegression(y, x)`), применяется для моделирования вероятности события на основе одной или нескольких независимых переменных.
`categoricalInformationValue()`	Вычисляет информационную ценность для категориальных переменных (`categoricalInformationValue(variable, outcome)`), позволяет оценить значимость переменных в предсказании исхода.
`contingency()`	Вычисляет таблицу сопряженности для двух категориальных переменных (`contingency(variable1, variable2)`), используется для анализа взаимосвязи между переменными.
`cramersV()`	Вычисляет коэффициент Крамера для измерения ассоциации между двумя категориальными переменными (`cramersV(variable1, variable2)`), позволяет оценить силу связи между переменными.
`cramersVBiasCorrected()`	Вычисляет скорректированный на смещение коэффициент Крамера (`cramersVBiasCorrected(variable1, variable2)`), обеспечивает более точную оценку ассоциации при малых выборках.
`theilsU()`	Вычисляет индекс Тейла для измерения ассоциации между двумя категориальными переменными (`theilsU(variable1, variable2)`), предоставляет альтернативную метрику для оценки взаимосвязей.
`maxIntersections()`	Находит максимальное количество пересечений между наборами значений (`maxIntersections(set1, set2)`), используется для определения наибольшего перекрытия между группами.
`maxIntersectionsPosition()`	Находит позицию максимального количества пересечений (`maxIntersectionsPosition(set1, set2)`), позволяет определить, где происходит наибольшее перекрытие между наборами данных.
`meanZTest()`	Выполняет Z-тест для среднего значения (`meanZTest(sampleMean, populationMean, stdDev, sampleSize)`), применяется для проверки гипотезы о равенстве средних в генеральной совокупности и выборке.
`sparkBar()`	Генерирует текстовое представление гистограммы для распределения значений (`sparkBar(column)`), предоставляет визуализацию распределения в текстовом формате.
`sumCount()`	Вычисляет сумму и количество значений одновременно (`sumCount(column)`), полезно для получения общей суммы и размера выборки в одном запросе.

Функция	Описание
`largestTriangleThreeBuckets()`	Применяет алгоритм Largest Triangle Three Buckets для уменьшения числа точек во временных рядах (`largestTriangleThreeBuckets(time, value, buckets)`), используется для оптимизации отображения графиков больших временных рядов.

Условные функции

Условные функции в RQL обеспечивают выполнение логических и сравнительных операций, позволяя определить выполнение различных выражений в зависимости от заданных условий.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список условных функций для ClickHouse 24.7.

Таблица 17. Условные функции

Функция	Описание
`if(cond, then, else)`	Выполняет условное ветвление. Если условие `cond` истинно (не равно нулю), возвращает результат выражения `then`. Если `cond` ложно (равно нулю) или NULL, возвращает результат выражения `else`. Типы выражений `then` и `else` должны быть схожими.
`multiIf(cond_1, then_1, cond_2, then_2, ⋯, else)`	Позволяет компактно записать несколько условий. Функция принимает 2N+1 аргументов: условия `cond_N` и соответствующие результаты `then_N`, а также результат `else` для случая, когда ни одно из условий не выполнено.

Функции сравнения

Типы данных, которые могут быть сравнены:

- Числа;
- Строки и фиксированные строки;
- Даты;
- Даты со временем.

Только значения в рамках одной группы могут быть сравнены (например, `UInt16` и `UInt64`), но не между группами (например, `UInt16` и `DateTime`).

Строки сравниваются по байтам. Если одна из строк содержит символы в кодировке UTF-8, занимающие несколько байт, это может привести к неожиданным результатам.

Строка `S1`, которая имеет другую строку `S2` в качестве префикса, считается длиннее, чем `S2`.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций сравнения для ClickHouse 24.7.

Таблица 18. Функции сравнения

Функция	Описание
`equals`, `=`, `==`	Проверяет равенство двух значений. Возвращает 1 (истина), если значения равны, и 0 (ложь) в противном случае.
`notEquals`, `!=`, `<>`	Проверяет, не равны ли друг другу два значения. Возвращает 1 (истина), если значения не равны, и 0 (ложь), если равны.
`less`, `<`	Проверяет, меньше ли первое значение, чем второе. Возвращает 1 (истина), если первое значение меньше второго, и 0 (ложь) в противном случае.
`greater`, `>`	Проверяет, больше ли первое значение, чем второе. Возвращает 1 (истина), если первое значение больше второго, и 0 (ложь) в противном случае.
`lessOrEquals`, `<=`	Проверяет, меньше ли первое значение, чем второе, или равны. Возвращает 1 (истина), если первое значение меньше второго или равно ему, и 0 (ложь) в противном случае.
`greaterOrEquals`, `>=`	Проверяет, больше ли первое значение, чем второе, или равны. Возвращает 1 (истина), если первое значение больше второго или равно ему, и 0 (ложь) в противном случае.
`greatest(⋯)`	Возвращает наибольшее значение из списка значений. Все элементы списка должны быть сопоставимыми типами. Тип возвращаемого значения может быть продвинут до более общего типа для сравнения.
`least(⋯)`	Возвращает наименьшее значение из списка значений. Аналогично `greatest`, все элементы списка должны быть сопоставимыми типами, и тип возвращаемого значения может быть продвинут.

Логические функции

Нижеперечисленные функции выполняют логические операции над аргументами произвольных числовых типов. Они возвращают либо 0, либо 1 типа `UInt8` или в некоторых случаях `NULL`.

Ноль в качестве аргумента рассматривается как ложь, ненулевые значения считаются истиной.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список логических функций для ClickHouse 24.7.

Таблица 19. Логические функции

Функция	Описание
`and(val1, val2⋯)`	Вычисляет результат логической конъюнкции (логического "И") между двумя или более значениями. Возвращает 1, если все аргументы истинны (ненулевые), 0 — если хотя бы один аргумент ложен (ноль), и NULL, если среди аргументов нет нулей, но есть хотя бы один NULL.
`or(val1, val2⋯)`	Вычисляет результат логической дизъюнкции (логического "ИЛИ") между двумя или более значениями. Возвращает 1, если хотя бы один аргумент истинен (ненулевой), 0 — если все аргументы ложны (нули), и NULL, если среди аргументов нет ненулевых значений и есть NULL.
`not(val)`	Вычисляет логическое отрицание (НЕ) аргумента. Возвращает 1, если аргумент ложен (ноль), 0 — если аргумент истинен (ненулевой), и NULL, если аргумент NULL.
`xor(val1, val2⋯)`	Вычисляет результат логической исключающей дизъюнкции (XOR) между двумя или более значениями. Для пары значений возвращает 1, если одно из значений истинно (ненулевое), а другое ложно (ноль). Возвращает 0, если оба значения одинаковы (оба нули или оба ненулевые). Для более чем двух значений применяется рекурсивно. Возвращает NULL, если среди аргументов хотя бы один NULL.

Функции преобразования типов

В RQL поведение функций `to<type>` и `cast` в некоторых случаях различается, например, при работе с `LowCardinality`: функция `cast` убирает признак `LowCardinality`, в то время как функции `to<type>` его не убирают. То же самое касается `Nullable`. Такое поведение не соответствует стандарту SQL, но его можно изменить с помощью настройки `cast_keep_nullable`.

Возможна потеря данных при конвертации значений из типа данных в меньший тип данных (например, с `Int64` на `Int32`) или между несовместимыми типами данных (например, с `String` на `Int`). Убедитесь, что результат конвертации соответствует ожиданиям.

Функции преобразования типов в RQL позволяют преобразовывать значения из одного типа данных в другой.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций преобразования типов для ClickHouse 24.7.

Функции преобразования в заданные типы

Таблица 20. Функции преобразования в заданные типы

Функция	Описание
`reinterpretAs<Type>`	Преобразует значение в заданный тип без изменения битового представления значения.
`CAST(x, T)`	Явное преобразование значения `x` в тип `T`.
`accurateCast(x, T)`	Преобразует значение `x` в тип `T` с проверкой на переполнение и потерю данных.
`accurateCastOrNull(x, T)`	Преобразует значение `x` в тип `T` с проверкой на переполнение и потерю данных, возвращая NULL при неудаче.
`accurateCastOrDefault(x, T, default_value)`	Преобразует значение `x` в тип `T` с проверкой на переполнение и потерю данных, возвращая значение по умолчанию при неудаче.

Функции преобразования в численные типы

Имена функций, преобразующих значения в численные типы, имеют следующую структуру:

```
to<Type><Precision>[<ErrorHandling>]
```

Здесь:

- `<Type>` — численный тип;
- `<Precision>` — битность или точность числа;
- `<ErrorHandling>` — результат, возвращаемый при неудаче. Если поведение не указано, генерируется исключение.

Конкретные функции перечислены далее.

Таблица 21. Функции преобразования в целочисленные типы

Функция	Описание
`toInt8`	Преобразует значение в тип Int8. В случае неудачи генерируется исключение.
`toInt8OrZero`	Преобразует значение в тип Int8. В случае неудачи возвращает 0.
`toInt8OrNull`	Преобразует значение в тип Int8. В случае неудачи возвращает NULL.
`toInt8OrDefault`	Преобразует значение в тип Int8. В случае неудачи возвращает значение по умолчанию.
`toInt16`	Преобразует значение в тип Int16. В случае неудачи генерируется исключение.
`toInt16OrZero`	Преобразует значение в тип Int16. В случае неудачи возвращает 0.

Функция	Описание
toInt16OrNull	Преобразует значение в тип Int16. В случае неудачи возвращает NULL.
toInt16OrDefault	Преобразует значение в тип Int16. В случае неудачи возвращает значение по умолчанию.
toInt32	Преобразует значение в тип Int32. В случае неудачи генерируется исключение.
toInt32OrZero	Преобразует значение в тип Int32. В случае неудачи возвращает 0.
toInt32OrNull	Преобразует значение в тип Int32. В случае неудачи возвращает NULL.
toInt32OrDefault	Преобразует значение в тип Int32. В случае неудачи возвращает значение по умолчанию.
toInt64	Преобразует значение в тип Int64. В случае неудачи генерируется исключение.
toInt64OrZero	Преобразует значение в тип Int64. В случае неудачи возвращает 0.
toInt64OrNull	Преобразует значение в тип Int64. В случае неудачи возвращает NULL.
toInt64OrDefault	Преобразует значение в тип Int64. В случае неудачи возвращает значение по умолчанию.
toInt128	Преобразует значение в тип Int128. В случае неудачи генерируется исключение.
toInt128OrZero	Преобразует значение в тип Int128. В случае неудачи возвращает 0.
toInt128OrNull	Преобразует значение в тип Int128. В случае неудачи возвращает NULL.
toInt128OrDefault	Преобразует значение в тип Int128. В случае неудачи возвращает значение по умолчанию.
toInt256	Преобразует значение в тип Int256. В случае неудачи генерируется исключение.
toInt256OrZero	Преобразует значение в тип Int256. В случае неудачи возвращает 0.
toInt256OrNull	Преобразует значение в тип Int256. В случае неудачи возвращает NULL.
toInt256OrDefault	Преобразует значение в тип Int256. В случае неудачи возвращает значение по умолчанию.

Таблица 22. Функции преобразования в беззнаковые целочисленные типы

Функция	Описание
toUInt8	Преобразует значение в тип UInt8. В случае неудачи генерируется исключение.
toUInt8OrZero	Преобразует значение в тип UInt8. В случае неудачи возвращает 0.
toUInt8OrNull	Преобразует значение в тип UInt8. В случае неудачи возвращает NULL.
toUInt8OrDefault	Преобразует значение в тип UInt8. В случае неудачи возвращает значение по умолчанию.
toUInt16	Преобразует значение в тип UInt16. В случае неудачи генерируется исключение.
toUInt16OrZero	Преобразует значение в тип UInt16. В случае неудачи возвращает 0.
toUInt16OrNull	Преобразует значение в тип UInt16. В случае неудачи возвращает NULL.
toUInt16OrDefault	Преобразует значение в тип UInt16. В случае неудачи возвращает значение по умолчанию.
toUInt32	Преобразует значение в тип UInt32. В случае неудачи генерируется исключение.
toUInt32OrZero	Преобразует значение в тип UInt32. В случае неудачи возвращает 0.
toUInt32OrNull	Преобразует значение в тип UInt32. В случае неудачи возвращает NULL.
toUInt32OrDefault	Преобразует значение в тип UInt32. В случае неудачи возвращает значение по умолчанию.
toUInt64	Преобразует значение в тип UInt64. В случае неудачи генерируется исключение.
toUInt64OrZero	Преобразует значение в тип UInt64. В случае неудачи возвращает 0.
toUInt64OrNull	Преобразует значение в тип UInt64. В случае неудачи возвращает NULL.
toUInt64OrDefault	Преобразует значение в тип UInt64. В случае неудачи возвращает значение по умолчанию.
toUInt128	Преобразует значение в тип UInt128. В случае неудачи генерируется исключение.
toUInt128OrZero	Преобразует значение в тип UInt128. В случае неудачи возвращает 0.
toUInt128OrNull	Преобразует значение в тип UInt128. В случае неудачи возвращает NULL.
toUInt128OrDefault	Преобразует значение в тип UInt128. В случае неудачи возвращает значение по умолчанию.

Функция	Описание
toUInt256	Преобразует значение в тип UInt256. В случае неудачи генерируется исключение.
toUInt256OrZero	Преобразует значение в тип UInt256. В случае неудачи возвращает 0.
toUInt256OrNull	Преобразует значение в тип UInt256. В случае неудачи возвращает NULL.
toUInt256OrDefault	Преобразует значение в тип UInt256. В случае неудачи возвращает значение по умолчанию.

Таблица 23. Функции преобразования в типы с плавающей запятой

Функция	Описание
toFloat32	Преобразует значение в тип Float32. В случае неудачи генерируется исключение.
toFloat32OrZero	Преобразует значение в тип Float32. В случае неудачи возвращает 0.
toFloat32OrNull	Преобразует значение в тип Float32. В случае неудачи возвращает NULL.
toFloat32OrDefault	Преобразует значение в тип Float32. В случае неудачи возвращает значение по умолчанию.
toFloat64	Преобразует значение в тип Float64. В случае неудачи генерируется исключение.
toFloat64OrZero	Преобразует значение в тип Float64. В случае неудачи возвращает 0.
toFloat64OrNull	Преобразует значение в тип Float64. В случае неудачи возвращает NULL.
toFloat64OrDefault	Преобразует значение в тип Float64. В случае неудачи возвращает значение по умолчанию.

Таблица 24. Функции преобразования в десятичные типы

toDecimal32	**Преобразует значение в тип Decimal32. В случае неудачи генерируется исключение.**
toDecimal32OrZero	Преобразует значение в тип Decimal32. В случае неудачи возвращает 0.
toDecimal32OrNull	Преобразует значение в тип Decimal32. В случае неудачи возвращает NULL.
toDecimal32OrDefault	Преобразует значение в тип Decimal32. В случае неудачи возвращает значение по умолчанию.
toDecimal64	Преобразует значение в тип Decimal64. В случае неудачи генерируется исключение.

toDecimal32	**Преобразует значение в тип Decimal32. В случае неудачи генерируется исключение.**
toDecimal64OrZero	Преобразует значение в тип Decimal64. В случае неудачи возвращает 0.
toDecimal64OrNull	Преобразует значение в тип Decimal64. В случае неудачи возвращает NULL.
toDecimal64OrDefault	Преобразует значение в тип Decimal64. В случае неудачи возвращает значение по умолчанию.
toDecimal128	Преобразует значение в тип Decimal128. В случае неудачи генерируется исключение.
toDecimal128OrZero	Преобразует значение в тип Decimal128. В случае неудачи возвращает 0.
toDecimal128OrNull	Преобразует значение в тип Decimal128. В случае неудачи возвращает NULL.
toDecimal128OrDefault	Преобразует значение в тип Decimal128. В случае неудачи возвращает значение по умолчанию.
toDecimal256	Преобразует значение в тип Decimal256. В случае неудачи генерируется исключение.
toDecimal256OrZero	Преобразует значение в тип Decimal256. В случае неудачи возвращает 0.
toDecimal256OrNull	Преобразует значение в тип Decimal256. В случае неудачи возвращает NULL.
toDecimal256OrDefault	Преобразует значение в тип Decimal256. В случае неудачи возвращает значение по умолчанию.

Функции преобразования типов даты и времени

Таблица 25. Функции преобразования типов даты и времени

Функция	**Описание**
toDate	Преобразует значение в тип даты. В случае неудачи генерируется исключение.
toDateOrZero	Преобразует значение в тип даты. В случае неудачи возвращает '0000-00-00'.
toDateOrNull	Преобразует значение в тип даты. В случае неудачи возвращает NULL.
toDateOrDefault	Преобразует значение в тип даты. В случае неудачи возвращает значение по умолчанию.

Функция	Описание
toDateTime	Преобразует значение в тип даты со временем. В случае неудачи генерируется исключение.
toDateTimeOrZero	Преобразует значение в тип даты со временем. В случае неудачи возвращает '0000-00-00 00:00:00'.
toDateTimeOrNull	Преобразует значение в тип даты со временем. В случае неудачи возвращает NULL.
toDateTimeOrDefault	Преобразует значение в тип даты со временем. В случае неудачи возвращает значение по умолчанию.
toDate32	Преобразует значение в тип даты с расширенным диапазоном. В случае неудачи генерируется исключение.
toDate32OrZero	Преобразует значение в тип даты с расширенным диапазоном. В случае неудачи возвращает минимально возможную дату.
toDate32OrNull	Преобразует значение в тип даты с расширенным диапазоном. В случае неудачи возвращает NULL.
toDate32OrDefault	Преобразует значение в тип даты с расширенным диапазоном. В случае неудачи возвращает значение по умолчанию.
toDateTime64	Преобразует значение в тип даты со временем с высокой точностью. В случае неудачи генерируется исключение.
toDateTime64OrZero	Преобразует значение в тип даты со временем с высокой точностью. В случае неудачи возвращает '0000-00-00 00:00:00.000'.

Функция	Описание
`toDateTime64OrNull`	Преобразует значение в тип даты со временем с высокой точностью. В случае неудачи возвращает NULL.
`toDateTime64OrDefault`	Преобразует значение в тип даты со временем с высокой точностью. В случае неудачи возвращает значение по умолчанию.
`parseDateTimeBestEffort`	Пытается преобразовать строку в тип `DateTime`, распознавая множество форматов даты и времени, включая ISO 8601, RFC 1123, формат даты и времени ClickHouse, а также другие форматы. Поддерживает разбор строковых представлений Unix timestamp, различные строковые форматы даты и времени, а также строковые представления с указанием часового пояса.
`parseDateTime32BestEffort`	Аналогично `parseDateTimeBestEffort`, но преобразует строку в тип `DateTime32`.
`parseDateTimeBestEffortUS`	Ведет себя как `parseDateTimeBestEffort` для стандартных форматов даты, но также предпочитает американский формат даты (MM/DD/YYYY и подобные) в случае неоднозначности. Если месяц и день не могут быть однозначно идентифицированы, функция возвращается к поведению `parseDateTimeBestEffort`.
`parseDateTimeBestEffortOrNull`	Работает так же, как `parseDateTimeBestEffort`, но возвращает `NULL`, если строка не может быть преобразована в дату и время.

Функция	Описание
`parseDateTime32BestEffortOrNull`	Аналогично `parseDateTimeBestEffortOrNull`, но преобразует строку в тип `DateTime32`, возвращая `NULL` при неудаче.
`parseDateTimeBestEffortOrZero`	Работает так же, как `parseDateTimeBestEffort`, но возвращает нулевую дату и время (`1970-01-01 00:00:00`), если строка не может быть преобразована.
`parseDateTime32BestEffortOrZero`	Аналогично `parseDateTimeBestEffortOrZero`, но преобразует строку в тип `DateTime32`, возвращая нулевую дату и время при неудаче.
`parseDateTimeBestEffortUSOrNull`	Работает так же, как `parseDateTimeBestEffortUS`, но возвращает `NULL`, если строка не может быть преобразована в дату и время.
`parseDateTimeBestEffortUSOrZero`	Работает так же, как `parseDateTimeBestEffortUS`, но возвращает нулевую дату и время (`1970-01-01 00:00:00`), если строка не может быть преобразована.
`parseDateTime64BestEffort`	Работает так же, как `parseDateTimeBestEffort`, но также принимает миллисекунды и микросекунды, возвращая тип данных `DateTime64` с заданной точностью.
`parseDateTime64BestEffortUS`	Работает аналогично `parseDateTime64BestEffort`, но предполагает американский формат даты (MM/DD/YYYY и подобные) в случае неоднозначности.
`parseDateTime64BestEffortOrNull`	Работает так же, как `parseDateTime64BestEffort`, но возвращает `NULL`, если формат даты не может быть обработан.

Функция	Описание
`parseDateTime64BestEffortOrZero`	Работает так же, как `parseDateTime64BestEffort`, но возвращает нулевую дату и время (`1970-01-01 00:00:00.000`), если формат даты не может быть обработан.
`parseDateTime64BestEffortUSOrNull`	Работает аналогично `parseDateTime64BestEffortUS`, но возвращает `NULL`, если входная строка не может быть преобразована в тип данных `DateTime64`.
`parseDateTime64BestEffortUSOrZero`	Работает аналогично `parseDateTime64BestEffortUS`, но возвращает нулевую дату и время (`1970-01-01 00:00:00.000`), если входная строка не может быть преобразована.
`toLowCardinality(expr)`	Преобразует входные данные в версию `LowCardinality` того же типа данных, оптимизируя обработку и хранение данных за счет снижения количества уникальных значений.
`toUnixTimestamp64Milli(value)`	Преобразует значение `DateTime64` в значение `Int64`, представляющее количество миллисекунд с начала эпохи Unix (1970-01-01 00:00:00 UTC). Входное значение округляется соответствующим образом.
`toUnixTimestamp64Micro(value)`	Преобразует значение `DateTime64` в значение `Int64`, представляющее количество микросекунд с начала эпохи Unix. Входное значение округляется соответствующим образом.
`toUnixTimestamp64Nano(value)`	Преобразует значение `DateTime64` в значение `Int64`, представляющее количество наносекунд с начала эпохи Unix. Входное значение округляется соответствующим образом.

Функция	Описание
`fromUnixTimestamp64Milli(value [, timezone])`	Преобразует значение `Int64`, представляющее количество миллисекунд с начала эпохи Unix, в значение `DateTime64` с точностью до миллисекунд и опционально применяет часовой пояс.
`fromUnixTimestamp64Micro(value [, timezone])`	Преобразует значение `Int64`, представляющее количество микросекунд с начала эпохи Unix, в значение `DateTime64` с точностью до микросекунд и опционально применяет часовой пояс.
`fromUnixTimestamp64Nano(value [, timezone])`	Преобразует значение `Int64`, представляющее количество наносекунд с начала эпохи Unix, в значение `DateTime64` с точностью до наносекунд и опционально применяет часовой пояс.
`dateTimeToSnowflake(value)`	Преобразует значение `DateTime` в первый идентификатор Snowflake ID на текущий момент.
`dateTime64ToSnowflake(value)`	Преобразует значение `DateTime64` в первый идентификатор Snowflake ID на текущий момент.
`snowflakeToDateTime(value [, timezone])`	Извлекает время из Snowflake ID и преобразует его в формат `DateTime`, принимая во внимание часовой пояс, если он указан.
`snowflakeToDateTime64(value [, timezone])`	Извлекает время из Snowflake ID и преобразует его в формат `DateTime64`, принимая во внимание часовой пояс, если он указан.

Функции преобразования в интервалы времени

Таблица 26. Функции преобразования в интервалы времени

Функция	Описание
`toIntervalYear`	Преобразует числовое значение в тип IntervalYear.

Функция	Описание
`toIntervalQuarter`	Преобразует числовое значение в тип IntervalQuarter.
`toIntervalMonth`	Преобразует числовое значение в тип IntervalMonth.
`toIntervalWeek`	Преобразует числовое значение в тип IntervalWeek.
`toIntervalDay`	Преобразует числовое значение в тип IntervalDay.
`toIntervalHour`	Преобразует числовое значение в тип IntervalHour.
`toIntervalMinute`	Преобразует числовое значение в тип IntervalMinute.
`toIntervalSecond`	Преобразует числовое значение в тип IntervalSecond.

Таблица 27. Функции преобразования в строковые типы

Функция	Описание
`toString`	Преобразует значение в строковый тип. В случае неудачи генерируется исключение.
`toFixedString(s, N)`	Преобразует строку в фиксированную строку заданной длины N. Если строка короче, она дополняется нулевыми байтами.
`toStringCutToZero(s)`	Обрезает строку до первого нулевого байта.
`toDecimalString(number, scale)`	Преобразует числовое значение в строку с указанным числом знаков после десятичной точки.
`formatRow(format, x, y, ⋯)`	Преобразует произвольные выражения в строку заданного формата, например, `CSV` или `TSV`, и может включать префикс/суффикс в каждой строке в зависимости от настроек.
`formatRowNoNewline(format, x, y, ⋯)`	Аналогично функции `formatRow`, но удаляет завершающий перевод строки в конце отформатированной строки, если он есть.

Функции для работы с Nullable-аргументами

Функции для работы с Nullable-аргументами в RQL предназначены для управления и обработки данных, которые могут содержать `NULL` значения.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse,

выбрав нужную версию из веток. Например: список функций для работы с Nullable-аргументами для ClickHouse 24.7.

Таблица 28. Функции для работы с Nullable-аргументами

Функция	Описание
`isNull(x)`	Проверяет, является ли аргумент `x` `NULL`. Возвращает `1`, если `x` — `NULL`, и `0`, если `x` — не `NULL`. Синоним: `ISNULL`.
`isNotNull(x)`	Проверяет, не является ли аргумент `x` `NULL`. Возвращает `0`, если `x` — `NULL`, и `1`, если `x` — не `NULL`.
`coalesce(x, ⋯)`	Последовательно слева направо проверяет, являются ли переданные аргументы `NULL`, и возвращает первый не `NULL` аргумент. Если все аргументы — `NULL`, возвращает `NULL`.
`ifNull(x, alt)`	Возвращает альтернативное значение `alt`, если основной аргумент `x` — `NULL`. В противном случае возвращает значение `x`.
`nullIf(x, y)`	Возвращает `NULL`, если аргументы `x` и `y` равны. В противном случае возвращает `x`.
`assumeNotNull(x)`	Приводит значение типа `Nullable` к не `Nullable`, если значение не `NULL`. Возвращает неспецифицированный результат, если исходное значение — `NULL`.
`toNullable(x)`	Преобразует тип аргумента `x` к `Nullable`. Возвращает входное значение с типом `Nullable`.

Математические функции

Все функции возвращают числа типа `Float64`.

Арифметические функции рассмотрены отдельно.

Точность получаемых результатов приближена к максимально возможной. Полученные значения могут не совпадать точно с ближайшими к ним числами, представимыми в машинном представлении.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список математических функций для ClickHouse 24.7.

Функции математических констант и проверки знака

Функция	Описание
`e()`	Возвращает число *e* (основание натурального логарифма), приблизительно равное 2.718.

Функция	Описание
`pi()`	Возвращает число π ("пи"), приблизительно равное 3.14159.
`sign(x)`	Возвращает знак числа x: -1, 0 или 1.

Функции возведения в степень и извлечения корня

Функция	Описание
`exp(x)`	Возвращает значение e в степени x.
`exp2(x)`	Возвращает 2 в степени x.
`intExp2(x)`	Аналог `exp` с возвращением значения UInt64.
`exp10(x)`	Возвращает 10 в степени x.
`intExp10(x)`	Аналог `exp10`, но возвращает UInt64.
`pow(x, y)`	Возвращает x в степени y.
`sqrt(x)`	Возвращает квадратный корень из x.
`cbrt(x)`	Возвращает кубический корень из x.

Логарифмические функции

Функция	Описание
`log(x)`	Возвращает натуральный логарифм x.
`log2(x)`	Возвращает двоичный логарифм x.
`log1p(x)`	Возвращает логарифм (1 + x).
`log10(x)`	Возвращает десятичный логарифм x.

Тригонометрические функции

Функция	Описание
`sin(x)`	Возвращает синус угла x, заданного в радианах.
`cos(x)`	Возвращает косинус угла x, заданного в радианах.
`tan(x)`	Возвращает тангенс угла x, заданного в радианах.
`asin(x)`	Возвращает арксинус x.
`acos(x)`	Возвращает арккосинус x.
`atan(x)`	Возвращает арктангенс x.
`cosh(x)`	Возвращает гиперболический косинус угла x.
`acosh(x)`	Возвращает обратный гиперболический косинус x.
`sinh(x)`	Возвращает гиперболический синус угла x.
`asinh(x)`	Возвращает обратный гиперболический синус x.
`atanh(x)`	Возвращает обратный гиперболический тангенс x.

Функция	Описание
`atan2(y, x)`	Возвращает арктангенс двух переменных y и x.
`hypot(x, y)`	Возвращает длину гипотенузы прямоугольного треугольника с катетами x и y.
`degrees(x)`	Конвертирует радианы в градусы.
`radians(x)`	Конвертирует градусы в радианы.

Статистические функции

Функция	Описание
`erf(x)`	Возвращает функцию ошибок для x.
`erfc(x)`	Возвращает дополнительную функцию ошибок для x.
`width_bucket(operand, low, high, count)`	Возвращает номер интервала, в который попадает значение operand, в гистограмме с равномерно распределенными интервалами от low до high.

Гамма-функция и факториал

Функция	Описание
`lgamma(x)`	Возвращает натуральный логарифм гамма-функции от x.
`tgamma(x)`	Возвращает гамма-функцию от x.
`factorial(n)`	Возвращает факториал n.

Арифметические функции

Для всех арифметических функций в RQL тип результата вычисляется как минимальный числовой тип, который может вместить результат, если такой тип существует. Выбор минимального типа происходит с учетом числа бит, знаковости и "плавучести". Если минимальных бит не хватает, то используется тип с максимальной битностью.

Примеры использования:

```
SELECT toTypeName(0), toTypeName(0 + 0), toTypeName(0 + 0 + 0), toTypeName(0 + 0 + 0 + 0)
```

Визуализация ожидаемого результата запроса:

toTypeName(0)	toTypeName(plus(0, 0))	toTypeName(plus(plus(0, 0), 0))	toTypeName(plus(plus(plus(0, 0), 0), 0))
UInt8	UInt16	UInt32	UInt64

Арифметические функции работают для любой пары типов из UInt8, UInt16, UInt32, UInt64, Int8, Int16, Int32, Int64, Float32, Float64. Переполнение обрабатывается аналогично C++.

Использование функций `multiplyDecimal` и `divideDecimal` предпочтительнее в ситуациях, когда необходима высокая точность вычислений и контроль за размером дробной части. В то же время следует учитывать, что эти функции могут работать медленнее стандартных арифметических функций `multiply` и `divide`.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список арифметических функций для ClickHouse 24.7.

Таблица 29. Арифметические функции

Функция	Описание
`plus(a, b)`	Вычисляет сумму чисел `a` и `b`. Поддерживает сложение целых чисел с датой и датой с временем, где прибавление к дате означает дни, а к дате с временем — секунды.
`minus(a, b)`	Вычисляет разность чисел `a` и `b`. Поддерживает вычитание целых чисел из даты и даты с временем, аналогично `plus`.
`multiply(a, b)`	Вычисляет произведение чисел `a` и `b`.
`divide(a, b)`	Вычисляет частное чисел `a` и `b`. Результат всегда с плавающей запятой. Для целочисленного деления используйте `intDiv`.
`intDiv(a, b)`	Вычисляет целочисленное частное чисел `a` и `b` с округлением вниз.
`intDivOrZero(a, b)`	Аналог `intDiv`, но возвращает ноль при делении на ноль или при определенных условиях переполнения.
`modulo(a, b)`	Вычисляет остаток от деления числа `a` на `b`. Результат целочисленный для целых чисел и с плавающей точкой, если хотя бы один из аргументов таковым является.
`moduloOrZero(a, b)`	Аналог `modulo`, но возвращает ноль при делении на ноль.
`negate(a)`	Вычисляет число, обратное по знаку к `a`.
`abs(a)`	Возвращает абсолютное значение числа `a`. Для беззнаковых типов ничего не изменяет.
`gcd(a, b)`	Вычисляет наибольший общий делитель чисел `a` и `b`.
`lcm(a, b)`	Вычисляет наименьшее общее кратное чисел `a` и `b`.
`max2(value1, value2)`	Сравнивает числа `value1` и `value2`, возвращая максимальное значение. Результат приводится к `Float64`.
`min2(value1, value2)`	Сравнивает числа `value1` и `value2`, возвращая минимальное значение. Результат приводится к `Float64`.
`multiplyDecimal(a, b[, result_scale])`	Умножает два числа типа `Decimal`, позволяя контролировать размер дробной части результата через `result_scale`.

Функция	Описание
`divideDecimal(a, b[, result_scale])`	Делит два числа типа `Decimal`, позволяя контролировать размер дробной части результата через `result_scale`.

Битовые функции

Битовые функции работают для любой пары типов из `UInt8`, `UInt16`, `UInt32`, `UInt64`, `Int8`, `Int16`, `Int32`, `Int64`, `Float32`, `Float64`.

Тип результата — целое число, битность которого равна максимальной битности аргументов. Если хотя бы один аргумент знаковый, то результат — знаковое число. Если аргумент — число с плавающей запятой — оно приводится к Int64.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список битовых функций для ClickHouse 24.7.

Таблица 30. Битовые функции

Функция	Описание
`bitAnd(a, b)`	Выполняет побитовую операцию И (AND) над двумя числами `a` и `b`. Тип результата — целое число, битность которого равна максимальной битности аргументов.
`bitOr(a, b)`	Выполняет побитовую операцию ИЛИ (OR) над двумя числами `a` и `b`. Тип результата совпадает с типом входных значений.
`bitXor(a, b)`	Выполняет побитовую операцию исключающее ИЛИ (XOR) над двумя числами `a` и `b`. Тип результата совпадает с типом входных значений.
`bitNot(a)`	Выполняет побитовую операцию НЕ (NOT) над числом `a`, инвертируя каждый бит. Тип результата совпадает с типом входного значения.
`bitShiftLeft(a, b)`	Сдвигает бинарное представление числа `a` на `b` бит влево. Поддерживаются числа и строки. Тип результата совпадает с типом сдвигаемого значения.
`bitShiftRight(a, b)`	Сдвигает бинарное представление числа `a` на `b` бит вправо. Поддерживаются числа и строки. Тип результата совпадает с типом сдвигаемого значения.
`bitTest(number, index)`	Возвращает значение бита на позиции `index` в числе `number`. Отсчет начинается с 0 справа налево. Возвращает `1`, если бит установлен, и `0`, если бит сброшен.
`bitTestAll(number, index1, index2, ⋯)`	Проверяет, все ли указанные биты в позициях `index1, index2, ⋯` установлены в 1 в числе `number`. Возвращает `1`, если все указанные биты установлены, и `0` в противном случае.

Функция	Описание
`bitTestAny(number, index1, index2, ⋯)`	Проверяет, установлен ли хотя бы один из указанных битов в позициях `index1, index2, ⋯` в числе `number` в 1. Возвращает `1`, если хотя бы один указанный бит установлен, и `0` в противном случае.
`bitCount(x)`	Подсчитывает количество бит, установленных в 1, в числе `x`. Возвращает количество этих бит.
`bitHammingDistance(int1, int2)`	Возвращает расстояние Хэмминга между двумя целыми числами `int1` и `int2`, то есть количество позиций, в которых соответствующие биты различаются.

Функции округления

Функции округления в RQL предоставляют различные способы обработки числовых значений для достижения определенной степени точности или адаптации к специфическим требованиям аналитики.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций округления для ClickHouse 24.7.

Таблица 31. Функции округления

Функция	Описание
`floor(x[, N])`	Возвращает наибольшее круглое число, меньше или равно `x`. `N` определяет точность округления.
`ceil(x[, N])`, `ceiling(x[, N])`	Возвращает наименьшее круглое число, больше или равно `x`. `N` определяет точность округления.
`trunc(x[, N])`, `truncate(x[, N])`	Обрезает число `x` до `N` знаков после запятой без округления.
`round(x[, N])`	Округляет число `x` до `N` знаков после запятой. При равном удалении от соседних чисел использует банковское округление для `Float32/64` и математическое для `Decimal`.
`roundBankers(x[, N])`	Округляет число `x` до `N` знаков после запятой, используя банковское округление для всех типов чисел.
`roundToExp2(num)`	Округляет число `num` вниз до ближайшей степени двух.
`roundDuration(num)`	Округляет число `num` до ближайшего значения из предопределенного набора чисел, обычно используемого для категоризации интервалов времени. Этот набор включает в себя следующие значения: 1, 10, 30, 60, 120, 180, 240, 300, 600, 1200, 1800, 3600, 7200, 18000, 36000.

Функция	Описание
`roundAge(num)`	Округляет число `num` до ближайшего значения из набора, представляющего стандартные возрастные группы. Этот метод позволяет классифицировать числовые значения возраста в следующие группы: 17 (для значений от 1 до 17), 18 (для значений от 18 до 24), 25 (для значений от 25 до 34), 35 (для значений от 35 до 44), 45 (для значений от 45 до 54) и 55 (для значений старше 55).
`roundDown(num, arr)`	Округляет число `num` вниз до ближайшего числа из предоставленного массива `arr`.

Функции для работы с датой и временем

Большинство функций в этом разделе принимают необязательный аргумент временной зоны, например, `Europe/Amsterdam`. В этом случае используется указанная временная зона, а не локальная (по умолчанию).

Пример

```
SELECT
    toDateTime('2016-06-15 23:00:00') AS time,
    toDate(time) AS date_local,
    toDate(time, 'Asia/Yekaterinburg') AS date_yekat,
    toString(time, 'US/Samoa') AS time_samoa
```

Визуализация ожидаемого результата запроса:

time	date_local	date_yekat	time_samoa
2016-06-15 23:00:00	2016-06-15	2016-06-16	2016-06-15 09:00:00

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций для работы с датой и временем для ClickHouse 24.7.

Таблица 32. Функции для работы с датой и временем

Функция	Описание
`makeDate`	Создает объект `Date` из года, месяца и дня или из года и дня года.
`makeDateTime`	Создает объект `DateTime` из года, месяца, дня, часа, минуты и секунды.
`timestamp`	Преобразует первый аргумент в тип `DateTime64(6)`. Если предоставлен второй аргумент, добавляет указанное время к преобразованному значению.
`timeZone`	Возвращает часовой пояс текущего сеанса.

Функция	Описание
serverTimeZone	Возвращает часовой пояс сервера.
toTimeZone	Конвертирует дату или дату со временем в указанный часовой пояс без изменения внутреннего значения.
timeZoneOf	Возвращает имя часового пояса для типов данных DateTime или DateTime64.
timeZoneOffset	Возвращает смещение часового пояса в секундах от UTC.
toYear	Возвращает годовой компонент (н.э.) даты или даты со временем.
toQuarter	Возвращает квартал (1-4) даты или даты со временем.
toMonth	Возвращает месячный компонент (1-12) даты или даты со временем.
toDayOfYear	Возвращает номер дня в году (1-366) даты или даты со временем.
toDayOfMonth	Возвращает номер дня в месяце (1-31) даты или даты со временем.
toDayOfWeek	Возвращает номер дня недели даты или даты со временем.
toHour	Возвращает часовой компонент (0-24) даты со временем.
toMinute	Возвращает минутный компонент (0-59) даты со временем.
toSecond	Возвращает секундный компонент (0-59) даты со временем. Секунды високосные не учитываются.
toMillisecond	Возвращает миллисекундный компонент (0-999) даты со временем.
toUnixTimestamp	Преобразует строку, дату или дату со временем в Unix Timestamp в представлении UInt32.
toStartOfYear	Округляет дату или дату со временем до первого дня года.
toStartOfISOYear	Округляет дату или дату со временем до первого дня ISO года, который может отличаться от "обычного" года.
toStartOfQuarter	Округляет дату или дату со временем до первого дня квартала.
toStartOfMonth	Округляет дату или дату со временем до первого дня месяца.
toLastDayOfMonth	Округляет дату или дату со временем до последнего дня месяца.
toMonday	Округляет дату или дату со временем до ближайшего понедельника.
toStartOfWeek	Округляет дату или дату со временем до начала недели, которая начинается в воскресенье или понедельник.
toLastDayOfWeek	Округляет дату или дату со временем до конца недели, которая заканчивается в субботу или воскресенье.
toStartOfDay	Округляет дату со временем до начала дня.
toStartOfHour	Округляет дату со временем до начала часа.
toStartOfMinute	Округляет дату со временем до начала минуты.
toStartOfSecond	Округляет дату со временем, удаляя подсекунды.
toStartOfFiveMinutes	Округляет дату со временем до начала пяти минутного интервала.
toStartOfTenMinutes	Округляет дату со временем до начала десяти минутного интервала.

Функция	Описание
`toStartOfFifteenMinutes`	Округляет дату со временем до начала пятнадцати минутного интервала.
`toStartOfInterval`	Обобщенная функция, аналогичная другим функциям `toStartOf*()`. Например, `toStartOfInterval(t, INTERVAL 1 year)` возвращает то же, что и `toStartOfYear(t)`.

Функции для работы со строками

В RQL представлен набор функций, предназначенных для обработки текстовых данных. Функции для поиска в строках, замены в строках, разбиения и слияния строк и массивов рассмотрены отдельно.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций для работы со строками для ClickHouse 24.7.

Функции проверки и приведения длины строк

Функция	Описание
`empty`	Проверяет, является ли входная строка пустой. Возвращает 1 для пустой строки или 0 для непустой строки.
`notEmpty`	Проверяет, не является ли входная строка пустой. Возвращает 1 для непустой строки или 0 для пустой строки.
`length`	Возвращает длину строки в байтах (а не символах или кодовых точках Unicode).
`lengthUTF8`	Возвращает длину строки в кодовых точках Unicode (а не в байтах или символах). Предполагается, что строка содержит допустимый текст в кодировке UTF-8.
`leftPad`	Дополняет строку слева пробелами или заданной строкой (при необходимости несколько раз), пока результирующая строка не достигнет указанной длины.
`leftPadUTF8`	Дополняет строку слева пробелами или заданной строкой (при необходимости несколько раз) до достижения указанной длины. В отличие от `leftPad`, длина строки измеряется в кодовых точках.
`rightPad`	Дополняет строку справа пробелами или заданной строкой (при необходимости несколько раз) до достижения указанной длины.
`rightPadUTF8`	Дополняет строку справа пробелами или заданной строкой (при необходимости несколько раз) до достижения указанной длины. В отличие от `rightPad`, длина строки измеряется в кодовых точках.
`trim`	Удаляет пробельные символы с начала и конца строки.
`trimLeft`	Удаляет пробельные символы с начала строки.

Функция	Описание
`trimRight`	Удаляет пробельные символы с конца строки.
`trimBoth`	Удаляет пробельные символы с обеих сторон строки.

Функции преобразования регистра

Функция	Описание
`lower`	Преобразует символы ASCII латиницы в строке в нижний регистр.
`upper`	Преобразует символы ASCII латиницы в строке в верхний регистр.
`lowerUTF8`	Преобразует строку в нижний регистр, предполагая, что строка содержит допустимый текст в кодировке UTF-8.
`upperUTF8`	Преобразует строку в верхний регистр, предполагая, что строка содержит допустимый текст в кодировке UTF-8.
`initcap`	Преобразует первый символ каждого слова в верхний регистр.
`initcapUTF8`	То же, что и `initcap`, но учитывает кодировку UTF-8.

Функции конкатенации и инверсии строк

Функция	Описание
`repeat`	Конкатенирует строку с самой собой указанное количество раз.
`space`	Конкатенирует пробел указанное количество раз с самим собой.
`concat`	Конкатенирует заданные аргументы.
`concatAssumeInjective`	Аналогично `concat`, но предполагает, что `concat(s1, s2, ⋯) → sn` является инъективной. Может использоваться для оптимизации `GROUP BY`.
`concatWithSeparator`	Конкатенирует заданные строки с использованием заданного разделителя.
`concatWithSeparatorAssumeInjective`	Аналогично `concatWithSeparator`, но предполагает, что `concatWithSeparator(sep, expr1, expr2, expr3⋯) → result` является инъективной. Может использоваться для оптимизации `GROUP BY`.
`reverse`	Инвертирует последовательность байтов в строке.
`reverseUTF8`	Инвертирует последовательность кодовых точек Unicode в строке. Предполагается, что строка содержит допустимый текст в кодировке UTF-8.
`appendTrailingCharIfAbsent`	Добавляет символ `c` к строке `s`, если `s` непустая и не заканчивается символом `c`.

Функции поиска и извлечения подстрок

Функция	Описание
`firstLine`	Возвращает первую строку из многострочного текста.

Функция	Описание
substring	Возвращает подстроку строки s, которая начинается с указанного индекса байта offset. Счет начинается с 1. Если offset равен 0, возвращается пустая строка. Если offset отрицательный, подстрока начинается с символов от конца строки.
substringUTF8	Аналогично substring, но для кодовых точек Unicode. Предполагается, что строка содержит допустимый текст в кодировке UTF-8.
substringIndex	Возвращает подстроку s до count вхождений разделителя delim, как в Spark или MySQL.
substringIndexUTF8	Аналогично substringIndex, но для кодовых точек Unicode. Предполагается, что строка содержит допустимый текст в кодировке UTF-8.
startsWith	Проверяет, начинается ли строка с определенного префикса.
startsWithUTF8	То же, что и startsWith, но с учетом кодировки UTF-8.
endsWith	Проверяет, заканчивается ли строка определенным суффиксом.
endsWithUTF8	То же, что и endsWith, но с учетом кодировки UTF-8.

Функции валидации, нормализации и кодирования строк

Функция	Описание
isValidUTF8	Возвращает 1, если набор байтов составляет допустимый текст, закодированный в UTF-8, в противном случае 0.
toValidUTF8	Заменяет недопустимые символы UTF-8 на символ (U+FFFD). Все подряд идущие недопустимые символы сжимаются в один символ замены.
convertCharset	Возвращает строку s, преобразованную из кодировки from в кодировку to.
base58Encode	Кодирует строку в формат Base58.
base58Decode	Декодирует строку из формата Base58.
tryBase58Decode	Попытка декодировать строку из формата Base58 с возвратом пустой строки в случае ошибки.
base64Encode	Кодирует строку в формат Base64.
base64Decode	Декодирует строку из формата Base64.
tryBase64Decode	Попытка декодировать строку из формата Base64 с возвратом пустой строки в случае ошибки.
normalizeUTF8NFC	Нормализует строку в нормализацию формы C (NFC) для Unicode.
normalizeUTF8NFD	Нормализует строку в нормализацию формы D (NFD) для Unicode.
normalizeUTF8NFKC	Нормализует строку в нормализацию формы KC (NFKC) для Unicode.
normalizeUTF8NFKD	Нормализует строку в нормализацию формы KD (NFKD) для Unicode.
encodeXMLComponent	Кодирует компоненты XML в строке.

Функция	Описание
decodeXMLComponent	Декодирует компоненты XML из строки.
decodeHTMLComponent	Декодирует HTML компоненты из строки.
extractTextFromHTML	Извлекает текст из HTML строки.
ascii	Возвращает ASCII код первого символа строки.
punycodeEncode	Кодирует строку в формат Punycode.
punycodeDecode	Декодирует строку из формата Punycode.
tryPunycodeDecode	Попытка декодировать строку из формата Punycode с возвратом пустой строки в случае ошибки.
idnaEncode	Кодирует строку в формат IDNA.
tryIdnaEncode	Попытка кодировать строку в формат IDNA с возвратом пустой строки в случае ошибки.
idnaDecode	Декодирует строку из формата IDNA.

Функции хеширования

Функция	Описание
CRC32	Вычисляет CRC32 хеш строки.
CRC32IEEE	Вычисляет CRC32 используя IEEE полином.
CRC64	Вычисляет CRC64 хеш строки.
normalizeQuery	Нормализует строку запроса.
normalizedQueryHash	Вычисляет хеш нормализованного запроса.

Функции оценки схожести строк

Функция	Описание
soundex	Преобразует строку в ее код Soundex.
byteHammingDistance	Вычисляет расстояние Хэмминга между двумя строками.
stringJaccardIndex	Вычисляет индекс Жаккара для двух строк.
stringJaccardIndexUTF8	Вычисляет индекс Жаккара для двух строк, учитывая кодировку UTF-8.
editDistance	Вычисляет расстояние редактирования (Левенштейна) между двумя строками.
damerauLevenshteinDistance	Вычисляет расстояние Дамерау-Левенштейна между двумя строками.
jaroSimilarity	Вычисляет сходство Джаро между двумя строками.
jaroWinklerSimilarity	Вычисляет сходство Джаро-Винклера между двумя строками.

Функции разбиения и слияния строк и массивов

Функции разбиения и слияния строк и массивов в RQL предназначены для манипуляции текстовыми данными и коллекциями. Эти функции позволяют разделять строки на подстроки или элементы

массива с использованием различных критериев, таких как фиксированные символы, строки, регулярные выражения или типы символов. Также они обеспечивают возможность объединения элементов массива в единую строку с заданным разделителем.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций разбиения и слияния строк и массивов для ClickHouse 24.7.

Таблица 33. Функции разбиения и слияния строк и массивов

Функция	Описание
`splitByChar(separator, s)`	Разбивает строку `s` на массив подстрок, используя односимвольный разделитель `separator`.
`splitByString(separator, s)`	Разбивает строку `s` на массив подстрок, используя строку `separator` в качестве разделителя.
`splitByRegexp(regexp, s)`	Разбивает строку `s` на массив подстрок, используя регулярное выражение `regexp` в качестве разделителя.
`splitByWhitespace(s)`	Разбивает строку `s` на массив подстрок, используя пробельные символы в качестве разделителей.
`splitByNonAlpha(s)`	Разбивает строку `s` на массив подстрок, используя неалфавитные символы в качестве разделителей.
`arrayStringConcat(arr[, separator])`	Склеивает элементы массива `arr` в строку, используя `separator` в качестве разделителя.
`alphaTokens(s)`	Выделяет из строки `s` подстроки, состоящие из буквенных символов, и возвращает их в виде массива.
`ngrams(s, n)`	Разбивает строку `s` на n-граммы (последовательности из `n` символов) и возвращает их в виде массива.
`tokens(s)`	Разбивает строку `s` на токены, используя не буквенно-цифровые символы ASCII в качестве разделителей.

Функции для поиска в строках

Все функции в этом разделе по умолчанию осуществляют поиск с учетом регистра. Поиск без учета регистра обычно предоставляется отдельными вариантами функций. Обратите внимание, что поиск без учета регистра следует правилам перевода в нижний и верхний регистры английского языка. Например, прописной букве `i` в английском языке соответствует заглавная `I`, тогда как в турецком языке — заглавная `İ`. Результаты для языков, отличных от английского, могут быть неожиданными.

Функции в этом разделе также предполагают, что искомая строка и строка поиска кодируются текстом с однобайтовой кодировкой. В случае отклонения от этого предположения, система не генерирует исключений, и результат поиска остается неопределенным. Для работы со строками, закодированными в UTF-8, обычно предоставляются специализированные версии функций. Аналогично, использование функций, предназначенных для работы с UTF-8, с текстом, не

закодированным в UTF-8, не приводит к генерации исключений, и результаты также остаются неопределенными. Обратите внимание, что автоматическая нормализация Unicode не выполняется, для этого можно использовать функции `normalizeUTF8*()`.

Общие функции для работы со строками и функции для замены в строках описаны отдельно.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций для поиска в строках для ClickHouse 24.7.

Таблица 34. Функции поиска в строках

Функция	Описание
`position(haystack, needle[, start_pos])`	Возвращает позицию (в байтах, начиная с 1) подстроки `needle` в строке `haystack`.
`positionCaseInsensitive(haystack, needle[, start_pos])`	Поиск позиции подстроки в строке без учета регистра.
`positionUTF8(haystack, needle[, start_pos])`	Поиск позиции подстроки в UTF-8 строке.
`positionCaseInsensitiveUTF8(haystack, needle[, start_pos])`	Поиск позиции подстроки в UTF-8 строке без учета регистра.
`multiSearchAllPositions(haystack, [needle1, needle2, …, needleN])`	Возвращает массив позиций для множества подстрок в строке.
`multiSearchAllPositionsUTF8(haystack, [needle1, needle2, …, needleN])`	Поиск всех позиций множества подстрок в UTF-8 строке.
`multiSearchFirstPosition(haystack, [needle1, needle2, …, needleN])`	Возвращает наименьшую позицию среди всех указанных подстрок в строке.
`multiSearchFirstIndex(haystack, [needle1, needle2, …, needleN])`	Возвращает индекс первой найденной подстроки из списка в строке.
`multiSearchAny(haystack, [needle1, needle2, …, needleN])`	Возвращает 1, если хотя бы одна из указанных подстрок найдена в строке, иначе 0.
`match(haystack, pattern)`	Возвращает `true`, если строка соответствует регулярному выражению в синтаксисе регулярных выражений re2.
`multiMatchAny(haystack, [pattern1, pattern2, …, patternN])`	Проверяет соответствие строки хотя бы одному из регулярных выражений.
`multiMatchAnyIndex(haystack, [pattern1, pattern2, …, patternN])`	Возвращает индекс регулярного выражения, которому соответствует строка.

Функция	Описание
`multiMatchAllIndices(haystack, [pattern1, pattern2, ⋯, patternN])`	Возвращает массив индексов всех регулярных выражений, которым соответствует строка.
`multiFuzzyMatchAny(haystack, distance, [pattern1, pattern2, ⋯, patternN])`	Проверяет соответствие строки хотя бы одному из регулярных выражений с учетом допустимого расстояния редактирования.
`extract(haystack, pattern)`	Извлекает фрагмент строки, соответствующий регулярному выражению.
`extractAll(haystack, pattern)`	Извлекает все фрагменты строки, соответствующие регулярному выражению.
`extractAllGroupsHorizontal(haystack, pattern)`	Возвращает массив массивов, где каждый массив содержит фрагменты строки, соответствующие каждой группе в регулярном выражении.
`extractAllGroupsVertical(haystack, pattern)`	Возвращает массив массивов, где каждый внутренний массив содержит фрагменты, соответствующие группам в порядке их появления в строке.
`like(haystack, pattern)`	Возвращает `true`, если строка соответствует шаблону выражения `LIKE`.
`notLike(haystack, pattern)`	Возвращает `true`, если строка не соответствует шаблону выражения `LIKE`.
`ilike(haystack, pattern)`	Возвращает `true`, если строка соответствует шаблону выражения `LIKE` без учета регистра.
`notILike(haystack, pattern)`	Возвращает `true`, если строка не соответствует шаблону выражения `LIKE` без учета регистра.
`ngramDistance(haystack, needle)`	Вычисляет расстояние между строками на основе 4-грамм.
`ngramSearch(haystack, needle)`	Похоже на `ngramDistance`, но вычисляет несимметричную разницу между строками.
`countSubstrings(haystack, needle)`	Возвращает количество вхождений подстроки в строку.
`countMatches(haystack, pattern)`	Возвращает количество совпадений регулярного выражения в строке.
`regexpExtract(haystack, pattern[, index])`	Извлекает подстроку, соответствующую индексу группы в регулярном выражении.
`hasSubsequence(haystack, needle)`	Возвращает `1`, если подстрока является подпоследовательностью строки, иначе `0`.

Шаблон может включать обычные символы и следующие метасимволы:

- `%` — обозначает любое количество произвольных символов или отсутствие символов.
- `_` — обозначает один произвольный символ.
- `\` — используется для экранирования символов `%`, `_` и `\`.

Функции для замены строк

Функции для замены строк в RQL обеспечивают возможности поиска и замены подстрок в текстовых данных. Они позволяют изменять содержимое строк путем замены одних подстрок на другие, как на основе простого текстового сопоставления, так и с использованием регулярных выражений.

Функции для замены строк в RQL предоставляют возможности для модификации текстовых данных путем замены определенных подстрок или символов на другие. Эти функции обеспечивают простые и сложные замены, включая работу с регулярными выражениями и прямое отображение символов для точной настройки результатов замены.

Общие функции для работы со строками и функции для замены в строках описаны отдельно.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций для замены строк для ClickHouse 24.7.

Таблица 35. Функции для замены строк

Функция	Описание
`replaceOne(source, pattern, replacement)`	Заменяет первое вхождение подстроки `pattern` в строке `source` на строку `replacement`.
`replaceAll(source, pattern, replacement)`	Заменяет все вхождения подстроки `pattern` в строке `source` на строку `replacement`. Псевдоним: `replace`.
`replaceRegexpOne(source, pattern, replacement)`	Заменяет первое вхождение подстроки, соответствующей регулярному выражению `pattern` (в синтаксисе re2), в строке `source` на строку `replacement`. В `replacement` могут использоваться подстановки `\0-\9`, где `\1-\9` соответствуют 1-й по 9-й захваченной группе, а `\0` соответствует всему совпадению.
`replaceRegexpAll(source, pattern, replacement)`	Аналогично `replaceRegexpOne`, но заменяет все вхождения подстрок, соответствующих регулярному выражению `pattern`. Псевдоним: `REGEXP_REPLACE`.
`translate(s, from, to)`	Заменяет символы в строке `s`, используя прямое отображение символов, определенное строками `from` и `to`. Строки `from` и `to` должны быть константными ASCII строками одинакового размера. Не-ASCII символы в исходной строке не модифицируются.
`translateUTF8(string, from, to)`	Аналогично `translate`, но предполагает, что строки `s`, `from` и `to` закодированы в UTF-8.

Функции для работы с массивами

Функции для работы с массивами в RQL предоставляют набор операций для манипуляции массивами данных. Эти функции позволяют выполнять проверку на пустоту или заполненность массива, определять его длину, создавать пустые массивы различных типов, генерировать

числовые диапазоны, объединять массивы, извлекать элементы, а также проводить более сложные операции, такие как вычисление уникальности элементов, работу с подмассивами и манипуляцию порядком элементов.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций для работы с массивами для ClickHouse 24.7.

Таблица 36. Функции для работы с массивами

Функция	Описание
`empty(arr)`	Проверяет, является ли входной массив `arr` пустым. Массив считается пустым, если он не содержит ни одного элемента. Функция также поддерживает работу с типами `String` и `UUID`.
`notEmpty(arr)`	Проверяет, является ли входной массив `arr` непустым. Массив считается непустым, если он содержит хотя бы один элемент. Функция также поддерживает работу с типами `String` и `UUID`.
`length(arr)`	Возвращает количество элементов в массиве `arr`. Тип результата — `UInt64`. Функция также работает для строк. Можно оптимизировать работу функции с использованием настройки `optimize_functions_to_subcolumns`.
`emptyArrayUInt8()`	Возвращает пустой массив с элементами типа `UInt8`.
`emptyArrayUInt16()`	Возвращает пустой массив с элементами типа `UInt16`.
`emptyArrayUInt32()`	Возвращает пустой массив с элементами типа `UInt32`.
`emptyArrayUInt64()`	Возвращает пустой массив с элементами типа `UInt64`.
`emptyArrayInt8()`	Возвращает пустой массив с элементами типа `Int8`.
`emptyArrayInt16()`	Возвращает пустой массив с элементами типа `Int16`.
`emptyArrayInt32()`	Возвращает пустой массив с элементами типа `Int32`.
`emptyArrayInt64()`	Возвращает пустой массив с элементами типа `Int64`.
`emptyArrayFloat32()`	Возвращает пустой массив с элементами типа `Float32`.
`emptyArrayFloat64()`	Возвращает пустой массив с элементами типа `Float64`.
`emptyArrayDate()`	Возвращает пустой массив с элементами типа `Date`.
`emptyArrayDateTime()`	Возвращает пустой массив с элементами типа `DateTime`.
`emptyArrayString()`	Возвращает пустой массив строк.
`emptyArrayToSingle(arr)`	Принимает пустой массив `arr` и возвращает массив из одного элемента, равного значению по умолчанию для типа элементов `arr`.

Функция	Описание
`range([start, ] end [, step])`	Возвращает массив чисел от `start` до `end - 1` с шагом `step`. Аргументы `start` и `step` необязательны. По умолчанию `start = 0`, `step = 1`. Аргумент `end` обязателен. Если аргументы `start`, `end`, `step` имеют отрицательные значения или значение аргумента `end` меньше значения аргумента `start`, функция генерирует исключение. Если суммарная длина создаваемых массивов больше значения настройки `function_range_max_elements_in_block`, функция генерирует исключение.
`array(x1, ⋯), operator [x1, ⋯]`	Создает массив из аргументов функции. Аргументы должны быть константами и иметь типы, для которых есть наименьший общий тип. Должен быть передан хотя бы один аргумент, так как иначе непонятно, какого типа создавать массив. Возвращает результат типа `Array(T)`, где `T` — наименьший общий тип от переданных аргументов.
`arrayConcat(arrays)`	Объединяет массивы, переданные в качестве аргументов, в один массив. Пример: `SELECT arrayConcat([1, 2], [3, 4], [5, 6]) AS res` возвращает `[1,2,3,4,5,6]`.
`arrayElement(arr, n)`	Извлекает элемент с индексом `n` из массива `arr`. Индексы начинаются с единицы. Поддерживаются отрицательные индексы для доступа к элементам с конца массива. Если индекс выходит за границы массива, то возвращается значение по умолчанию для типа элементов массива.
`has(arr, elem)`	Проверяет наличие элемента `elem` в массиве `arr`. Возвращает `0`, если элемента в массиве нет, или `1`, если есть. `NULL` обрабатывается как значение. Пример: `SELECT has([1, 2, NULL], NULL)` возвращает `1`.
`indexOf(arr, x)`	Возвращает индекс первого вхождения элемента `x` в массив `arr`, начиная с `1`. Если элемент отсутствует в массиве, возвращает `0`. Элементы, равные `NULL`, обрабатываются как обычные значения.
`countEqual(arr, x)`	Возвращает количество элементов массива, равных `x`. Эквивалентно `arrayCount(elem → elem = x, arr)`. `NULL` обрабатывается как значение. Пример: `SELECT countEqual([1, 2, NULL, NULL], NULL)` возвращает `2`.
`arrayEnumerate(arr)`	Возвращает массив индексов для каждого элемента входного массива `arr`. Это позволяет идентифицировать позицию каждого элемента в исходном массиве.
`arrayEnumerateUniq(arr, ⋯)`	Возвращает массив, где каждый элемент указывает на его первое вхождение в массиве `arr`. Если функция принимает несколько массивов, уникальность определяется по комбинации элементов на соответствующих позициях во всех массивах.
`arrayPopBack(array)`	Удаляет последний элемент из массива `array` и возвращает новый массив без этого элемента.
`arrayPopFront(array)`	Удаляет первый элемент из массива `array` и возвращает новый массив без этого элемента.
`arrayPushBack(array, single_value)`	Добавляет элемент `single_value` в конец массива `array` и возвращает новый массив с добавленным элементом.

Функция	Описание
`arrayPushFront(array, single_value)`	Добавляет элемент `single_value` в начало массива `array` и возвращает новый массив с добавленным элементом.
`arrayResize(array, size[, extender])`	Изменяет размер массива `array` до нового размера `size`, используя `extender` для заполнения новых элементов, если `size` больше текущего размера массива.
`arraySlice(array, offset[, length])`	Возвращает часть массива `array`, начиная с позиции `offset` и длиной `length`. Если `length` не указано, возвращает все элементы начиная с `offset` до конца массива.
`arrayUniq(arr, ⋯)`	Возвращает массив уникальных элементов из массива `arr`. Если передано несколько массивов, возвращает массив уникальных комбинаций элементов на соответствующих позициях в этих массивах.
`arrayJoin(arr)`	Специальная функция, которая используется для разделения одного массива `arr` на строки, каждая из которых содержит один элемент массива.
`arrayDifference(array)`	Вычисляет разность между соседними элементами массива `array`. Возвращает массив, где первый элемент равен нулю, а каждый следующий элемент равен разности между текущим и предыдущим элементами исходного массива.
`arrayDistinct(array)`	Возвращает массив, содержащий только уникальные элементы из исходного массива `array`.
`arrayEnumerateDense(arr)`	Возвращает массив, каждый элемент которого указывает на позицию первого вхождения соответствующего элемента в массиве `arr` с учетом его встречаемости.
`arrayIntersect(arr)`	Возвращает массив, содержащий элементы, которые присутствуют во всех переданных массивах `arr`.
`arrayReverse(arr)`	Возвращает массив с элементами исходного массива `arr` в обратном порядке.
`reverse(arr)`	Синоним функции `arrayReverse`. Возвращает массив с элементами исходного массива `arr` в обратном порядке.
`arrayFlatten(array_of_arrays)`	Преобразует массив массивов `array_of_arrays` в один плоский массив, содержащий все элементы исходных массивов. Элементы вложенных массивов добавляются в результирующий массив последовательно, начиная с первого вложенного массива до последнего.
`arrayCompact(array)`	Удаляет последовательно повторяющиеся элементы из массива `array`, сохраняя только уникальные последовательности. Возвращает новый массив без последовательных дубликатов.
`arrayZip(arr1, arr2, ⋯)`	Объединяет несколько массивов `arr1`, `arr2`, ..., создавая массив кортежей, где каждый кортеж содержит элементы из исходных массивов, расположенные на одинаковых позициях. Длина результирующего массива соответствует длине самого короткого из исходных массивов.

Функция	Описание
`arrayAUC(arr_scores, arr_labels)`	Вычисляет площадь под кривой (AUC) для заданных массивов оценок `arr_scores` и меток `arr_labels`.
`arrayProduct(arr)`	Возвращает произведение элементов массива `arr`.
`arrayRotateLeft(arr, n)`	Циклически сдвигает элементы массива `arr` влево на `n` позиций.
`arrayRotateRight(arr, n)`	Циклически сдвигает элементы массива `arr` вправо на `n` позиций.
`arrayShiftLeft(arr, n)`	Сдвигает элементы массива `arr` влево на `n` позиций, заполняя освободившиеся места нулями.
`arrayShiftRight(arr, n)`	Сдвигает элементы массива `arr` вправо на `n` позиций, заполняя освободившиеся места нулями.

Географические функции

Функции географических координат

Функции географических координат в RQL предназначены для обработки и анализа геопространственных данных. Они позволяют выполнять расчеты расстояний между точками на земной поверхности, определять принадлежность точек к геометрическим фигурам, таким как эллипсы и многоугольники, и вычислять углы между точками.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций географических координат для ClickHouse 24.7.

Функция	Описание
`greatCircleDistance`	Расчет расстояния по большому кругу между двумя точками на поверхности земли.
`geoDistance`	Вычисляет расстояние между двумя точками на земной поверхности.
`greatCircleAngle`	Вычисляет угол между двумя точками на поверхности земли по большому кругу.
`pointInEllipses`	Определяет, находится ли точка внутри заданных эллипсов.
`pointInPolygon`	Проверяет, находится ли заданная точка внутри многоугольника.

Функции Geohash

Geohash — это система геокодирования, которая делит поверхность Земли на сектора в форме сетки и кодирует каждую ячейку короткой строкой из букв и цифр. Это иерархическая структура данных, поэтому чем длиннее строка geohash, тем точнее географическое местоположение.

Если вам необходимо вручную конвертировать географические координаты в строки geohash, вы можете использовать geohash.org.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций Geohash для ClickHouse 24.7.

Функция	Описание
`geohashEncode`	Кодирует географические координаты в строку Geohash.
`geohashDecode`	Декодирует строку Geohash обратно в географические координаты.
`geohashesInBox`	Генерирует Geohash-коды, покрывающие прямоугольник, заданный двумя угловыми точками.

Функции индексов H3

H3 — это географическая система индексации, в которой поверхность Земли разделена на сетку равных шестиугольных ячеек. Эта система является иерархической, то есть каждый шестиугольник на верхнем уровне ("родитель") может быть разделен на семь равных, но меньших ("дети"), и так далее.

Уровень иерархии называется разрешением и может принимать значения от 0 до 15, где 0 — это базовый уровень с самыми крупными и грубыми ячейками.

Пара широты и долготы может быть преобразована в 64-битный индекс H3, идентифицирующий ячейку сетки.

Индекс H3 в основном используется для группировки местоположений и других геопространственных манипуляций.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций индексов H3 для ClickHouse 24.7.

Функция	Описание
`h3IsValid`	Проверяет, является ли заданный индекс H3 действительным.
`h3GetResolution`	Возвращает разрешение (уровень детализации) индекса H3.
`h3EdgeAngle`	Вычисляет угол ребра индекса H3.
`h3EdgeLengthM`	Вычисляет длину ребра индекса H3 в метрах.
`h3EdgeLengthKm`	Вычисляет длину ребра индекса H3 в километрах.
`geoToH3`	Преобразует географические координаты в индекс H3.
`h3ToGeo`	Преобразует индекс H3 обратно в географические координаты.

Функция	Описание
h3ToGeoBoundary	Возвращает границы гексагона, представленного индексом H3.
h3kRing	Возвращает набор индексов H3, составляющих кольцо вокруг заданной точки.
h3GetBaseCell	Возвращает базовую ячейку для данного индекса H3.
h3HexAreaM2	Возвращает площадь гексагона, представленного индексом H3, в квадратных метрах.
h3HexAreaKm2	Возвращает площадь гексагона, представленного индексом H3, в квадратных километрах.
h3IndexesAreNeighbors	Проверяет, являются ли два индекса H3 соседями.
h3ToChildren	Возвращает дочерние индексы для заданного индекса H3 и указанного разрешения.
h3ToParent	Возвращает родительский индекс для заданного индекса H3 и указанного разрешения.
h3ToString	Преобразует индекс H3 в строковое представление.
stringToH3	Преобразует строковое представление индекса H3 обратно в индекс.
h3IsResClassIII	Определяет, принадлежит ли индекс H3 к третьему классу разрешения.
h3IsPentagon	Проверяет, является ли индекс H3 пентагоном.
h3GetFaces	Возвращает список граней, которые пересекает гексагон, представленный индексом H3.
h3CellAreaM2	Возвращает площадь ячейки H3 в квадратных метрах.
h3CellAreaRads2	Возвращает площадь ячейки H3 в радианах^2.
h3ToCenterChild	Возвращает центральный дочерний индекс для заданного индекса H3.
h3ExactEdgeLengthM	Возвращает точную длину ребра индекса H3 в метрах.
h3ExactEdgeLengthKm	Возвращает точную длину ребра индекса H3 в километрах.
h3ExactEdgeLengthRads	Возвращает точную длину ребра индекса H3 в радианах.
h3NumHexagons	Возвращает количество гексагонов в указанном разрешении.
h3Line	Возвращает линию (последовательность индексов H3) между двумя индексами H3.
h3Distance	Возвращает расстояние в гексагонах между двумя индексами H3.
h3HexRing	Возвращает кольцо гексагонов на заданном расстоянии от центрального индекса H3.
h3GetUnidirectionalEdge	Возвращает однонаправленное ребро между двумя индексами H3.
h3UnidirectionalEdgeIsValid	Проверяет, является ли однонаправленное ребро действительным.
h3GetOriginIndexFromUnidirectionalEdge	Возвращает исходный индекс для однонаправленного ребра.
h3GetDestinationIndexFromUnidirectionalEdge	Возвращает конечный индекс для однонаправленного ребра.

Функция	Описание
`h3GetIndexesFromUnidirectionalEdge`	Возвращает оба индекса для однонаправленного ребра.
`h3GetUnidirectionalEdgesFromHexagon`	Возвращает все однонаправленные ребра для заданного гексагона.
`h3GetUnidirectionalEdgeBoundary`	Возвращает границы однонаправленного ребра.

Функции индексов S2

S2 — это система географической индексации, в которой все географические данные представлены на трехмерной сфере (аналогично глобусу).

В библиотеке S2 точки представлены как S2 Index — определенное число, которое внутренне кодирует точку на поверхности единичной сферы, в отличие от традиционных пар (широта, долгота). Для получения индекса точки S2 для данной точки, указанной в формате (широта, долгота), используйте функцию geoToS2. Также вы можете использовать функцию s2ToGeo для получения географических координат, соответствующих указанному индексу точки S2.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций индексов S2 для ClickHouse 24.7.

Функция	Описание
`geoToS2`	Преобразует географические координаты в индекс S2.
`s2ToGeo`	Преобразует индекс S2 обратно в географические координаты.
`s2GetNeighbors`	Возвращает индексы S2 соседних ячеек.
`s2CellsIntersect`	Проверяет пересечение двух ячеек S2.
`s2CapContains`	Проверяет, содержит ли сферический "cap" заданную точку или ячейку.
`s2CapUnion`	Объединяет два сферических "cap".
`s2RectAdd`	Добавляет точку к прямоугольнику S2, потенциально расширяя его.
`s2RectContains`	Проверяет, содержит ли прямоугольник S2 заданную точку.
`s2RectUnion`	Объединяет два прямоугольника S2.
`s2RectIntersection`	Находит пересечение двух прямоугольников S2.

Функции временных рядов

В RQL доступны специализированные функции для анализа и обработки данных временных рядов. Эти функции обеспечивают возможности для обнаружения выбросов, определения периодичности и декомпозиции временных рядов на составляющие, такие, как сезонность и тренд.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций временных рядов для ClickHouse 24.7.

Таблица 37. Функции временных рядов

Функция	Описание
`seriesOutliersDetectTukey(series, [min_percentile, max_percentile, K])`	Обнаруживает выбросы в данных временных рядов, используя ограды Тьюки. Принимает массив числовых значений `series`, минимальный и максимальный процентили `min_percentile` и `max_percentile` для расчета межквартильного размаха (IQR) и константу `K` для обнаружения умеренных или более сильных выбросов. По умолчанию `min_percentile` = 0.25, `max_percentile` = 0.75, `K` = 1.5. Требуется минимум четыре точки данных в `series` для обнаружения выбросов. Возвращает массив той же длины, что и входной, где каждое значение представляет собой оценку возможной аномалии соответствующего элемента в ряду.
`seriesPeriodDetectFFT(series)`	Находит период данных временного ряда с использованием быстрого преобразования Фурье (FFT). Принимает массив числовых значений `series`. Возвращает действительное значение, равное периоду данных временного ряда, или NAN, если количество точек данных меньше четырех.
`seriesDecomposeSTL(series, period)`	Декомпозиция временного ряда на сезонную, трендовую и остаточную компоненты с использованием STL (Seasonal and Trend decomposition using Loess). Принимает массив числовых значений `series` и положительное целое число `period`, указывающее период временного ряда. Количество точек данных в `series` должно быть как минимум в два раза больше значения `period`. Возвращает массив из четырех массивов, где первый массив включает сезонные компоненты, второй — тренд, третий — остаточную компоненту, и четвертый — базовую компоненту (сезонная + тренд).

Функции для работы с IP-адресами

Функции для работы с IP-адресами в RQL предназначены для выполнения различных операций преобразования и анализа с IPv4 и IPv6 адресами.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций для работы с IP-адресами для ClickHouse 24.7.

Таблица 38. Функции для работы с IP-адресами

Функция	Описание
`IPv4NumToString(num)`	Принимает число `UInt32`, интерпретирует его как IPv4-адрес в `big-endian` и возвращает строку с соответствующим IPv4-адресом в формате `A.B.C.D`.
`IPv4StringToNum(s)`	Обратная функция к `IPv4NumToString`. В случае неверного формата IPv4-адреса генерирует исключение.
`IPv4StringToNumOrDefault(s)`	Аналогично `IPv4StringToNum`, но возвращает 0 при неверном формате IPv4-адреса.
`IPv4StringToNumOrNull(s)`	Аналогично `IPv4StringToNum`, но возвращает `null` при неверном формате IPv4-адреса.
`IPv4NumToStringClassC(num)`	Похожа на `IPv4NumToString`, но заменяет последний октет на xxx.
`IPv6NumToString(x)`	Принимает значение `FixedString(16)`, содержащее IPv6-адрес в бинарном формате, и возвращает строку с этим адресом в текстовом формате.
`IPv6StringToNum(s)`	Обратная функция к `IPv6NumToString`. В случае неверного формата IPv6-адреса генерирует исключение.
`IPv6StringToNumOrDefault(s)`	Аналогично `IPv6StringToNum`, но возвращает 0 при неверном формате IPv6-адреса.
`IPv6StringToNumOrNull(s)`	Аналогично `IPv6StringToNum`, но возвращает `null` при неверном формате IPv6-адреса.
`IPv4ToIPv6(x)`	Принимает число `UInt32`, интерпретирует его как IPv4-адрес в `big-endian` и возвращает значение `FixedString(16)`, содержащее IPv6-адрес в бинарном формате.
`cutIPv6(x, bytesToCutForIPv6, bytesToCutForIPv4)`	Принимает значение `FixedString(16)`, содержащее IPv6-адрес в бинарном формате, и возвращает строку с адресом с убранным указанным количеством байт.
`IPv4CIDRToRange(ipv4, Cidr)`	Принимает IPv4 и значение `UInt8`, содержащее CIDR, и возвращает кортеж из двух IPv4, содержащих нижний и верхний диапазон сети.
`IPv6CIDRToRange(ipv6, Cidr)`	Принимает IPv6 и значение `UInt8`, содержащее `CIDR`, и возвращает кортеж из двух IPv6, содержащих нижний и верхний диапазон сети.
`toIPv4(string)`	Является псевдонимом`IPv4StringToNum()`, принимает строковую форму IPv4-адреса и возвращает значение типа IPv4, бинарно равное значению, возвращаемому `IPv4StringToNum()`.
`toIPv4OrDefault(string)`	Аналогично `toIPv4`, но возвращает 0.0.0.0 (IPv4 типа 0) при неверном формате IPv4-адреса.
`toIPv4OrNull(string)`	Аналогично `toIPv4`, но возвращает `null` при неверном формате IPv4-адреса.
`toIPv6(string)`	Конвертирует строковую форму IPv6-адреса в тип IPv6. В случае неверного формата IPv6-адреса возвращает пустое значение.
`toIPv6OrDefault(string)`	Аналогично `toIPv6`, но возвращает `::` (IPv6 типа 0) при неверном формате IPv6-адреса.

Функция	Описание
`toIPv6OrNull(string)`	Аналогично `toIPv6`, но возвращает `null` при неверном формате IPv6-адреса.
`isIPv4String`	Определяет, является ли входная строка IPv4-адресом. Возвращает 1, если строка является IPv4-адресом, и 0 в противном случае.
`isIPv6String`	Определяет, является ли входная строка IPv6-адресом. Возвращает 1, если строка является IPv6-адресом, и 0 в противном случае.
`isIPAddressInRange`	Определяет, содержится ли IP-адрес в сети, представленной в нотации CIDR. Возвращает 1, если это так, и 0 в противном случае.

Функции для работы с URL

Функции для работы с URL предназначены для анализа и манипуляции URL-адресами. Они позволяют извлекать компоненты URL, такие, как протокол, домен, порт, путь, параметры запроса, а также проводить различные операции по преобразованию и очистке URL-адресов.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций для работы с URL для ClickHouse 24.7.

Таблица 39. Функции для работы с URL

Функция	Описание
`protocol(URL)`	Возвращает протокол из URL.
`domain(URL)`	Извлекает имя хоста из URL.
`domainWithoutWWW(URL)`	Возвращает домен, удалив префикс `www.`, если он присутствовал.
`topLevelDomain(URL)`	Извлекает домен верхнего уровня из URL.
`firstSignificantSubdomain(URL)`	Возвращает первый существенный поддомен из URL.
`cutToFirstSignificantSubdomain(URL)`	Возвращает часть домена до первого существенного поддомена.
`cutToFirstSignificantSubdomainCustom(URL, TLD)`	Возвращает часть домена до первого существенного поддомена с использованием пользовательского списка доменов верхнего уровня.
`cutToFirstSignificantSubdomainCustomWithWWW(URL, TLD)`	То же самое, но не удаляет `www`.
`firstSignificantSubdomainCustom(URL, TLD)`	Возвращает первый существенный поддомен с использованием пользовательского списка доменов верхнего уровня.
`port(URL[, default_port = 0])`	Возвращает порт из URL или значение `default_port`, если порт не указан.
`path(URL)`	Возвращает путь из URL.

Функция	Описание
`pathFull(URL)`	Возвращает путь, включая query string и fragment из URL.
`queryString(URL)`	Возвращает query string из URL.
`fragment(URL)`	Возвращает fragment из URL.
`queryStringAndFragment(URL)`	Возвращает query string и fragment из URL.
`extractURLParameter(URL, name)`	Извлекает значение параметра `name` из URL.
`extractURLParameters(URL)`	Возвращает массив параметров из URL.
`extractURLParameterNames(URL)`	Возвращает массив имен параметров из URL.
`URLHierarchy(URL)`	Возвращает массив с URL, обрезанный по `/`, `?`.
`URLPathHierarchy(URL)`	Возвращает массив путей из URL, обрезанный по `/`.
`decodeURLComponent(URL)`	Возвращает декодированный URL.
`netloc(URL)`	Извлекает сетевую локальность (`username:password@host:port`) из URL.
`cutWWW(URL)`	Удаляет `www.` из начала домена в URL.
`cutQueryString(URL)`	Удаляет query string из URL.
`cutFragment(URL)`	Удаляет fragment из URL.
`cutQueryStringAndFragment(URL)`	Удаляет query string и fragment из URL.
`cutURLParameter(URL, name)`	Удаляет параметр с именем `name` из URL.

Функции для работы с UUID

Функции для работы с UUID обеспечивают создание, преобразование и проверку уникальных идентификаторов, применяемых для обеспечения уникальности объектов в базах данных, системах и приложениях. Эти функции позволяют генерировать UUID четвертой версии, проверять UUID на пустоту или наличие значений, а также осуществлять преобразования между строковыми и двоичными представлениями UUID. Кроме того, доступна функция для получения уникального идентификатора сервера, что может быть использовано для идентификации источника данных или обеспечения трассировки в распределенных системах.

Набор поддерживаемых функций RQL и их поведение зависят от версии установленного ClickHouse. Если функция не поддерживается в используемой версии ClickHouse, она также не будет доступна в RQL. Ознакомиться с актуальным списком функций и их поведением можно в официальном репозитории ClickHouse, выбрав нужную версию из веток. Например: список функций для работы с UUID для ClickHouse 24.7.

Таблица 40. Функции для работы с UUID

Функция	Описание
`generateUUIDv4([x])`	Генерирует UUID четвертой версии. Необязательный параметр `x` используется для обхода удаления общих подвыражений, если функция вызывается несколько раз в одном запросе. Возвращает значение типа UUID.
`empty(UUID)`	Проверяет, является ли входной UUID пустым (содержит все нули). Работает также с массивами и строками. Возвращает 1 для пустого UUID и 0 для непустого.
`notEmpty(UUID)`	Проверяет, не является ли входной UUID пустым. Возвращает 1 для непустого UUID и 0 для пустого.
`UUIDStringToNum(uuid[, variant = 1])`	Преобразует строковое представление UUID в двоичное, с указанием варианта формата (по умолчанию big-endian). Возвращает значение типа `FixedString(16)`.
`UUIDNumToString(binary[, variant = 1])`	Принимает двоичное представление UUID и возвращает строку из 36 символов в текстовом формате. Вариант формата может быть указан (по умолчанию Big-endian).
`serverUUID()`	Возвращает случайный и уникальный UUID сервера, который генерируется при первом запуске и сохраняется навсегда. Возвращает UUID сервера.

Справочник по языку преобразования данных VRL

В данном разделе представлен справочник по языку преобразования данных VRL (Vector Remap Language), который применяется в схемах RObject для трансформации, фильтрации и обогащения данных событий в системе.

Содержание раздела:

- О языке VRL
- Выражения
- Функции
  - Функции разбора
  - Функции кодирования и декодирования
  - Функции событий и активных списков
  - Функции коллекций
  - Функции строк
  - Функции типов и форматирования
  - Хэш-функции и криптографические функции
  - Функции генерации случайных значений и ID
  - Системные функции
  - Функции отладки
- Справочник ошибок
- Советы по написанию кода на языке VRL

О языке VRL

Vector Remap Language (VRL) — язык на основе выражений, созданный для работы с данными наблюдения. В него встроены различные функции, предназначенные для обработки журналов и метрик.

Применение VRL для преобразований данных

VRL работает на базе инструмента обработки данных Vector. Изначально в Vector преобразование данных осуществлялось на основе встроенного языка конфигурации, и появление VRL позволило существенно сократить описание трансформаций.

Рассмотрим распространенный случай использования — преобразование общего журнала Apache, поступающего из Docker.

```
{
  "time": "2021-02-03T21:13:54.713161211Z",
```

```
  "stream": "stdout",
  "log": "198.51.100.12 - user [03/Feb/2021:21:13:55 -0200] \"GET /embrace/supply-
chains/dynamic/vertical HTTP/1.0\" 201 20574"
}
```

Анализ этого журнала без VRL в Vector выглядит следующим образом:

```
# ... источники ...
# Анализ внутреннего журнала Syslog.
[transforms.parse_syslog]
type = "regex_parser"
inputs = ["parse_docker"]
patterns = ['^(?P<host>\S+) (?P<client>\S+) (?P<user>\S+) \[(?<timestamp>[\w:/]+\s[+\-
]\d{4})\] "(?<method>\S+) (?<resource>.+?) (?<protocol>\S+)" (?<status>\d{3})
(?<bytes_out>\S+)$']
field = "log"

# Удаление дублирующихся полей времени и журнала Docker.

[transform.remove_log]
type = "remove_fields"
inputs = ["parse_syslog"]
fields = ["time", "log"]

# Принудительная обработка полей.
[transforms.coerce_fields]
type = "coercer"
inputs = ["remove_log"]
types.timestamp = "timestamp"
types.status = "int"
types.bytes_out = "int"

# Приемники данных.
...
```

Результат преобразования:

```
{
  "bytes_out": 20574,
  "host": "198.51.100.12",
  "method": "GET",
  "resource": "/embrace/supply-chains/dynamic/vertical",
  "protocol": "HTTP/1.0",
  "status": 201,
  "timestamp": "2021-02-03T23:13:55Z",
  "client": "-",
  "user": "user"
}
```

С помощью языка VRL можно проанализировать вышеуказанный фрагмент так:

```
. = parse_common_log!(.log)
.total_bytes = del(.size)
.internal_request = ip_cidr_contains("198.51.100.0/16", .host) ?? false
```

Результат преобразования:

```
{
  "host": "198.51.100.12",
  "internal_request": true,
  "user": "user",
  "timestamp": "2021-02-03T23:13:55Z",
  "message": "GET /embrace/supply-chains/dynamic/vertical HTTP/1.0",
  "method": "GET",
  "path": "/embrace/supply-chains/dynamic/vertical",
  "protocol": "HTTP/1.0",
  "total_bytes": 20574,
  "status": 201
}
```

Конфигурация преобразования на VRL проще в написании, чтении и управлении, чем аналогичная на Vector. Добавлено дополнительное поле `internal_request`, чтобы продемонстрировать возможности встроенных функций VRL.

Принципы и характеристики языка VRL

VRL разработан с использованием принципов безопасности и производительности, оставаясь при этом гибким, поэтому он подходит для задач высокопроизводительной обработки событий ИБ.

Типобезопасность и надежность

VRL реализует принципы типобезопасности и надежности путем явного контроля над обработкой возможных ошибок. Это можно проиллюстрировать на примере фрагмента, который упоминался выше.

```
{
  "time":"2021-02-03T21:13:54.713161211Z",
  "stream": "stdout",
  "log": "198.51.100.12 - user [03/Feb/2021:21:13:55 -0200] \"GET /embrace/supply-
chains/dynamic/vertical HTTP/1.0\" 201 20574"
}
```

Его необходимо проанализировать и привести к результату:

```
{
  "host": "198.51.100.12",
```

```
  "user": "user",
  "timestamp": "2021-02-03T23:13:55Z",
  "message": "GET /embrace/supply-chains/dynamic/vertical HTTP/1.0",
  "method": "GET",
  "path": "/embrace/supply-chains/dynamic/vertical",
  "protocol": "HTTP/1.0",
  "total_bytes": 20574,
  "status": 201
}
```

Пользователь, только начавший работать с VRL, может написать следующую программу:

```
. = parse_common_log(.log)
.total_bytes = del(.size)
```

При компиляции он получит следующее сообщение об ошибке (Vector):

```
error[E103]: unhandled fallible assignment
┌── :1:5
│
1 │ . = parse_common_log(.log)
│ ___ ^^^^^^^^^^^^^^^^^^^^^^^
│ │   │
│ │   this expression is fallible
│ │   update the expression to be infallible
│ or change this to an infallible assignment:
│ ., err = parse_common_log(.log)
│
= see documentation about error handling at https://errors.vrl.dev/#handling
= learn more about error code 103 at https://errors.vrl.dev/103
= see language documentation at https://vrl.dev
```

Как видно, VRL требует от пользователя обработки любого выражения, которое может привести к ошибке во время выполнения. В нашем случае `.log` может не быть строкой, поэтому нужно либо указать тип поля `.log`, либо обработать ошибку в случае, если `.log` не является строкой. Чтобы устранить эту ошибку, пользователь должен выполнить одно из следующих действий:

- Обработка ошибки:

  ```
  ., err = parse_common_log(.log)
  if err != null {
    .malformed = true
    log("Failed to parse common-log: " + err, level: "error")
  } else {
    .total_bytes = del(.size)
  }
  ```

Если событие имеет неправильную форму (`.log` не является отформатированной строкой common-log), мы не выводим ошибку в лог и добавляем поле `.malformed`. Это сохраняет исходные данные и позволяет легко направить искаженные данные на проверку. Часто обработкой этой ошибки пренебрегают, что приводит к потере данных и простою системы.

- Выявление ошибки и прерывание программы:

```
. = parse_common_log!(.log)
.total_bytes = del(.size)
```

Иногда наличие искаженных данных бывает неприемлемым, и программа должна быть прервана. В этом случае VRL предлагает варианты ошибочных функций. Здесь указывается, что добавление к вызову функции постфикса `!` вызовет сбой и прервет выполнение программы. При этом Vector перейдет к обработке следующего события, но одновременно запишет в журнал сообщение об ошибке. Таким образом, пользователям предлагается выбрать метод обработки ошибок.

- Указание типов:

```
.log = to_string!(.log)

., err = parse_common_log(.log)
if err != null {
  # This error only occurs for malformed *strings*.
  log("Failed to parse common-log: " + err, level: "error")
} else {
  .total_bytes = del(.size)
}
```

В приведенных примерах можно заметить, что ошибку типа трудно отличить от ошибки синтаксического анализа. Возможно, вы предпочтете прерывать выполнение программы при возникновении ошибок типа и обрабатывать ошибки синтаксического анализа. Это достигается с помощью **прогрессивной безопасности типов**. В ходе оценки программа на языке VRL накапливает информацию о типах ваших полей. Когда тип распознается, ошибки типа исключаются при последующем использовании этого поля. Таким образом, в приведенном выше примере пользователь знает, что ошибка может возникнуть только в случае сбоя синтаксического анализа.

Эргономическая безопасность

VRL обеспечивает эргономическую безопасность, то есть делает сложным создание медленных или нестабильных программ. Проверки, выполняемые во время компиляции, предотвращают ошибки во время выполнения. Однако они не могут выявить некоторые более тонкие проблемы с производительностью и поддерживаемостью, которые возникают из-за сложности программы. Эти проблемы могут привести к нестабильности системы сбора и анализа метрик, а также непредвиденным затратам ресурсов.

Для защиты от эргономических трудностей, VRL был строго ограничен, предлагая только те функции, которые необходимы для преобразования данных наблюдения. Любые функции, которые не

соответствуют данной задаче или могут ухудшить эргономику, были исключены из языка.

Безопасность доступа к памяти

VRL наследует от Rust гарантии безопасности доступа к памяти. Это позволяет избежать распространенных программных ошибок и уязвимостей, которые возникают из-за неправильного обращения к памяти. Благодаря этому VRL подходит для инфраструктурных приложений, таких как конвейеры обработки событий, где надежность и безопасность являются первостепенными целями.

Выражения

Синтаксис

В данной статье при описании выражений приняты следующие обозначения:

- Элементы выражений разделены тильдой.
- Круглые скобки служат для группировки элементов выражения.
- Если в выражении используется литерал скобок или других символов, они заключаются в кавычки: `"("`. `"NEWLINE"` обозначает перевод строки.
- Квантификаторы после элементов и групп обозначают количество повторений:
  - `?` — элемент необязателен, может отсутствовать в выражении;
  - `+` — элемент обязателен и может повторяться в выражении несколько раз;
  - `*` — элемент может отсутствовать или повторяться несколько раз.

Комментарий

Комментарий используется для целей документирования программного кода и обозначается символом `#`. Каждая строка комментария должна начинаться с символа `#`. В настоящее время VRL не поддерживает многострочные комментарии.

Примеры

```
# комментарий
```

```
# комментарий
# из нескольких строк
```

Ключевые слова

Ключевые слова — это зарезервированные слова, используемые для базовых функций языка, таких как работа с условиями (`if`). Они не могут использоваться для наименования переменных или других пользовательских директив. Следующие слова зарезервированы:

- `abort`

- `as`
- `break`
- `continue`
- `else`
- `false`
- `for`
- `if`
- `impl`
- `in`
- `let`
- `loop`
- `null`
- `return`
- `self`
- `std`
- `then`
- `this`
- `true`
- `type`
- `until`
- `use`
- `while`

Пробелы

Пробел — это любая непустая строка, определенная свойством Unicode `White_Space`.

VRL является языком свободной формы, что означает, что все пробелы и отступы используются только для разделения токенов в грамматике и не влияют на семантику программы.

Литералы

Пустое значение (Null)

Литерал *пустого значения* представляет отсутствие определенного значения.

Пример

```
null
```

Логическое значение (Boolean)

Литерал типа *логическое выражение* представляет собой двоичное значение, которое может быть только истинным (`true`) или ложным (`false`).

Примеры

```
true
```

```
false
```

Целое число (Integer)

Литерал типа *целое число* — это десятичное представление 64-битного целого числа со знаком.

Характеристики

Ограничения

Целые числа в VRL могут находиться в диапазоне от $-2^{63} + 1$ до $2^{63} - 1$. Числа за пределами этого диапазона "оборачиваются", то есть переносятся на другой конец диапазона.

Подчеркивание

В целых числах может использоваться символ `_` для улучшения читаемости, например, для разделения разрядов. Например, `1_000_000`.

Примеры

```
1_000_000
```

```
1000000
```

Число с плавающей точкой (Float)

Литерал типа *число с плавающей точкой* представляет собой десятичное представление 64-битного числа с плавающей точкой (в частности, тип "binary64", определенный в IEEE 754-2008), включая знак.

Десятичный литерал с плавающей точкой состоит из целой части (десятичные цифры), десятичной точки и дробной части (десятичные цифры).

Характеристики

Ограничения

Числа с плавающей точкой в VRL могут находиться в диапазоне от `-1.7976931348623157E+308f64` до `1.7976931348623157E+308f64`. Числа за пределами этого диапазона "оборачиваются", то есть переносятся на другой конец диапазона.

Подчеркивания

В числах с плавающей точкой может использоваться символ `_` для улучшения читаемости, например, для разделения разрядов. Например, `1_000_000.0`.

Примеры

```
1_000_000.01
```

```
1000000.01
```

```
1.001
```

Метка времени (Timestamp)

Литерал метки времени — это нативная метка времени, выраженная в формате RFC 3339 с точностью до наносекунды.

Метки времени определяются с помощью символа `t` и заключаются в одинарные кавычки (`t'2021-02-11T10:32:50.553955473Z'`).

Примеры

```
t'2021-02-11T10:32:50.553955473Z'
```

```
t'2021-02-11T10:32:50.553Z'
```

```
t'2021-02-11T10:32:50.553-04:00'
```

Характеристики

Часовые пояса

Как определено в формате RFC 3339, литералы меток времени поддерживают UTC и локальные смещения времени.

Строка (String)

Литерал строки представляет собой строку, закодированную в UTF-8. Строковые литералы могут быть сырыми или интерпретированными.

Сырые (raw) **строковые литералы** состоят из неинтерпретированных (неявно закодированных в UTF-8) символов между одинарными кавычками и обозначаются символом `s` (`s'…'`). В частности, обратная косая черта не имеет особого значения, и строка может содержать символы новой строки.

Интерпретированные строковые литералы представляют собой последовательности символов между двойными кавычками (`"···"`). Внутри кавычек может находиться любой символ, кроме неэкранированного символа новой строки и неэкранированной двойной кавычки. Текст между кавычками представляет собой результат литерала, с экранированными символами, интерпретируемыми как показано ниже. Строки можно шаблонизировать, заключив переменные в `{{..}}`. Значения переменных вставляются в строку на указанной позиции.

Примеры

```
"Hello, world! 🌎"
```

```
"Hello, world! \u1F30E"
```

```
"Hello, \
world!"
```

```
"Hello, {{ planet }}!"
```

```
s'Hello, world!'
```

```
s'{ "foo": "bar" }'
```

Характеристики

Экранирование

Специальные символы, такие как символы новой строки, могут быть выражены с помощью экранирования обратной косой чертой.

Представление	Описание
`\u{7FFF}`	Символ Unicode из 24 бит (до 6 цифр)
`\n`	Перевод строки
`\r`	Возврат каретки
`\t`	Табуляция
`\\`	Обратная косая черта
\0	Пустой символ
`\"`	Двойные кавычки
`\'`	Одинарные кавычки
\{	Фигурная скобка

Шаблоны

Строку можно шаблонизировать, заключив имя переменной в `{{..}}`. Значение переменной вставляется в строку на этой позиции во время выполнения. Переменная должна быть строкой. Поддерживаются только переменные. Если вы хотите вставить путь из события, вы должны сначала присвоить его переменной. Чтобы вставить двойные фигурные скобки в строку, их можно экранировать с помощью символа `\`: `\{{..\}}`.

Многострочные литералы

Длинные строки можно разбить на несколько строк, добавив обратную косую черту перед символом новой строки. Символ новой строки и любые пробелы в начале следующей строки не будут включены в строку.

Конкатенация

Строки можно объединять с помощью оператора `+`.

Недопустимые символы

Недопустимые последовательности UTF-8 заменяются символом .

Регулярное выражение (Regex)

Литерал *регулярного выражения* представляет собой регулярное выражение, используемое для сопоставления и разбора строк.

Регулярные выражения определяются с помощью символа `r` и заключаются в одинарные кавычки: `r'…'`. Используется синтаксис регулярных выражений Rust.

Для изучения регулярных выражений в VRL рекомендуется использовать редактор и тестер выражений Rustexp.

Обычные и специальные символы

Представление	Описание	Эквиваленты

`a`	Большинство символов представляют сами себя.	`\x61`
		`\u0061`
		`\U00000061`

`\\`	Специальные символы `[ ] \ / ^ $ . | ? * + ( ) { }` для представления самих себя должны быть экранированы.	`\x5C`
`\x61`	Шестнадцатеричный код символа ASCII из двух цифр.	`a`
`\u0062`	Шестнадцатеричный код символа Unicode из четырех цифр.	`b`
`\U00000062`	Шестнадцатеричный код символа Unicode из восьми цифр.	`c`

Представление	Описание	Эквиваленты

`\x{00003A9}`
`\u{000003A9}`
`\U{3A9}`
	Любой шестнадцатеричный код, которому соответствует символ Unicode.	`Ω`

`\t`	Символ горизонтальной табуляции.	`\x09`
`\n`	Символ перевода строки	`\x0A`
`\r`	Символ возврата каретки.	`\x0D`
`\v`	Символ вертикальной табуляции.	`\x0B`
`\a`	Символ звукового сигнала.	`\x07`
`\f`	Символ смены страницы.	`\x0C`

Метасимволы и специальные символьные классы

Представление	Описание	Эквиваленты
`.`	Метасимвол, обозначающий любой символ, кроме символа перевода строки `\n`.	
`\p{Property}`	Любой символ со свойством Unicode, указанным в фигурных скобках. Список поддерживаемых свойств описан в документе.	
`\P{Property}`	Любой символ, не имеющий свойства Unicode, указанного в фигурных скобках.	
`\pX`	Любой символ Unicode, относящийся к классу, обозначенному одной буквой: • `L` — "алфавитные" символы: буквы и идеограммы всех письменностей; • `M` — диакритические символы; • `N` — цифры; • `P` — знаки препинания; • `S` — знаки валют, математические и иные обозначения; • `Z` — пробельные символы, разделители строк и абзацев; • `C` — управляющие символы, символы из областей для частного использования и нераспределенных областей.	`\p{Property}`
`\PX`	Любой символ Unicode, не относящийся к классу, обозначенному одной буквой.	`\P{Property}`
`[[:ascii:]]`	Любой символ ASCII.	`[\x00-\x7F]`

Представление	Описание	Эквиваленты
`[[:print:]]`	Любой символ ASCII, доступный для печати: любой символ, кроме управляющих.	`[ -~]`
`[[:graph:]]`	Любой отображаемый символ ASCII: любой символ, кроме управляющих и пробельных.	`[!-~]`
`\w`	Любой символ Unicode, входящий в состав слов: буква или идеограмма любой письменности, диакритический символ, цифра, знак пунктуации, соединяющий слова (например, _), управляющий символ для лигатур.	`[\p{Alphabetic}\p{M}\d\p{Pc}\p{Join_Control}]`
`\W`	Любой символ Unicode, не входящий в состав слов.	`[^\w]`

`[[:word:]]`	Любая буква базового латинского алфавита, цифра ASCII или подчеркивание.	`[0-9A-Za-z_]`
		`(?-u:\w)`
		`[\w&&\p{ascii}]`

`[[:alnum:]]`	Любая буква базового латинского алфавита или цифра ASCII.	`[0-9A-Za-z]`
`[[:alpha:]]`	Любая буква базового латинского алфавита.	`[A-Za-z]`
`[[:upper:]]`	Любая прописная буква базового латинского алфавита.	`[A-Z]`
`[[:lower:]]`	Любая строчная буква базового латинского алфавита.	`[a-z]`
`\d`	Любая десятичная цифра Unicode.	`\p{Nd}`
`\D`	Любой символ, кроме десятичных цифр Unicode.	`[^\d]`

`[[:digit:]]`	Любая десятичная цифра ASCII.	`[0-9]`
		`[\d&&\p{ascii}]`
		`(?-u:\d)`

`[[:xdigit:]]`	Любая шестнадцатеричная цифра.	`[0-9A-Fa-f]`
`[[:punct:]]`	Любой знак препинания ASCII.	`[!-/:@\\\[\]\{\}\(\)~‘]`
`\s`	Любой пробельный символ (символ со свойством Unicode `Whitespace`).	`\p{White_Space}`
`\S`	Любой символ, кроме пробельных.	`\P{White_Space}`
`[[:space:]]`	Любой пробельный символ ASCII.	`[\t\n\v\f\r ]`
`[[:blank:]]`	Пробел или горизонтальная табуляция.	`[\t ]`
`[[:cntrl:]]`	Любой управляющий символ ASCII.	`[\x00-\x1F\x7F]`

Символьные классы и перечисление

Представление	Описание	Найденные совпадения
`xy`	Последовательность `x` и `y`.	`xy`
`x|y`	Перечисление: одна из последовательностей `x` или `y`, причем предпочтение отдается `x`.	`x`
`[xyz]`	Объединение: символы из указанных в квадратных скобках.	`x`, `y`, `z`
`[^xyz]`	Отрицание: все символы, кроме указанных в квадратных скобках.	Все символы, кроме `x`, `y` и `z`.
`[a-z]`	Диапазон: все символы между указанными.	Символы от `a` до `z` включительно.
`[x[^xyz]]`	Вложенный класс: все символы, указанные хотя бы в одном из классов.	Все символы, кроме `y` и `z`.
`[a-y&&xyz]`	Пересечение: символы, которые входят в оба класса.	`x`, `y`
`[0-9--4]`	Разность: символы первого класса, которые не входят во второй класс. (Эквивалентно записи `[0-9&&[^4]]`).	Все десятичные цифры ASCII, кроме `4`.
`[a-g~~b-h]`	Симметрическая разность: символы, которые входят лишь в один класс из указанных.	`a`, `h`

Приоритет в перечислениях:

В перечислениях проверяется каждый указанный вариант слева направо, поиск прекращается при первом совпадении.

Пример

Пример 168. Исходный код

```
.res = parse_regex_all!("authentication", r'authentication|auth', numeric_groups:
true)
```

Пример 169. Результат

```
[
  {
    "0": "authentication"
  }
]
```

Здесь функция с шаблоном `r'authentication|auth'` сначала пытается найти последовательность `"authentication"`, и поскольку она полностью соответствует строке, именно это совпадение и будет возвращено. Однако, если изменить порядок вариантов, изменится и результат.

Пример 170. Исходный код

```
.res = parse_regex_all!("authentication", r'auth|authentication', numeric_groups:
true)
```

Пример 171. Результат

```
[
  {
    "0": "auth"
  }
]
```

При таком порядке вариантов сначала будет найдено `"auth"`. На этом поиск остановится и вернет `"auth"`, несмотря на то, что более длинный вариант `"authentication"` также соответствует строке.

Порядок вычисления символьных классов:

1. Диапазоны: выражение `r'[а-яё]'` равнозначно выражению `r'[[а-я]ё]'` и выбирает все строчные буквы русского алфавита.
2. Объединение: выражение `r'[ab&&bc]'` равнозначно выражению `r'[[ab]&&[bc]]'` и выбирает символы, которые входят в оба класса.
3. Пересечение, разность, симметрическая разность имеют одинаковый приоритет и вычисляются по порядку слева направо. Например, следующее выражение:

   ```
   r'[\pL--\p\{Greek}&&\p\{Uppercase}]'
   ```

   равнозначно такому выражению:

   ```
   r'[[\pL--\p\{Greek}]&&\p\{Uppercase}]'
   ```

   Эти выражения выбирают все "алфавитные" символы в верхнем регистре, кроме греческих.
4. Отрицание: выражение `r'[^a-z&&[:xdigit:]]'` равнозначно выражению `r'[^[a-z&&[:xdigit:]]]'`.

   Эти выражения выбирают все символы, кроме тех, которые одновременно являются шестнадцатеричными цифрами и строчными латинскими буквами.

Группировка

Группы в круглых скобках определяют области действия флагов, квантификаторов, и перечислений.

Регулярные выражения поддерживают захват групп, что позволяет извлекать части шаблона как строки. Именованные группы захвата позволяют связать результаты извлечения с ключами.

Представление	Описание
`(exp)`	Нумерованная группа захвата.

Представление	Описание
`(?P<name>exp)`	Именованная группа захвата. Возможно обращение как по номеру, так и по ключу, указанному в угловых скобках. (Эквивалентно записи `(?<name>exp)`.) Имя может включать "алфавитные" символы и цифры Unicode, а также символы `.,_[]` и начинаться с "алфавитного" символа либо подчеркивания.
`(?:exp)`	Незахватываемая группа.
`(?x)`	Установка флага `x` внутри текущей группы.
`(?-x)`	Выключение флага `x` внутри текущей группы.
`(?x-y:exp)`	Установка флага `x` и выключение флага `y` внутри незахватываемой группы.

Пример

Рассмотрим следующую функцию:

Пример 172. Исходный код

```
.res = parse_regex!("2012-12-12", r'(?P<y>\d{4})-(?P<m>\d{2})-(?P<d>\d{2})')
```

Пример 173. Результат

```
{
  "res": {
    "d": "12",
    "m": "12",
    "y": "2012"
  }
}
```

Данная функция извлекает из строки с датой вида `YYYY-MM-DD` группы, содержащие год, месяц и число с ключами `y`, `m` и `d` соответственно.

Квантификаторы

Квантификатор после символа, символьного класса или группы определяет, сколько раз может встречаться предшествующее выражение.

При жадной квантификации возвращается наиболее длинная строка из возможных соответствий, при ленивой — наиболее короткая.

Представление	Описание	Поведение
`x?`	Ноль или одно повторение `x`.	Жадная квантификация
`x*`	Ноль или больше повторений `x`.	Жадная квантификация

Представление	Описание	Поведение
`x+`	Одно или больше повторений `x`.	Жадная квантификация
`x??`	Ноль или одно повторение `x`.	Ленивая квантификация
`x*?`	Ноль или больше повторений `x`.	Ленивая квантификация
`x+?`	Одно или больше повторений `x`.	Ленивая квантификация
`x{n}`	В точности `n` повторений `x`. Аналогично записи `x{n}?`.	
`x{n,}`	Как минимум `n` повторений `x`.	Жадная квантификация
`x{n,m}`	Между `n` и `m` повторений `x`	Жадная квантификация
`x{n,}?`	Как минимум `n` повторений `x`.	Ленивая квантификация
`x{n,m}?`	Между `n` и `m` повторений `x`	Ленивая квантификация

Примеры

Следующая функция ищет последовательности символов, заключенные в квадратные скобки.

Пример 174. Исходный код

```
.res = parse_regex_all!("[ERROR] [2024-10-05 10:01:00] [code: 500]", r'\[.*\]',
numeric_groups: true)
```

Пример 175. Результат

```
[
  {
    "0": "[ERROR] [2024-10-05 10:01:00] [code: 500]"
  }
]
```

Если заменить квантификатор на ленивый: `r'\[.*?\]'`, функция вернет отдельные подстроки в квадратных скобках.

Пример 176. Исходный код

```
.res = parse_regex_all!("[ERROR] [2024-10-05 10:01:00] [code: 500]", r'\[.*?\]',
numeric_groups: true)
```

Пример 177. Результат

```
[
  {
    "0": "[ERROR]"
  },
  {
```

```
    "0": "[2024-10-05 10:01:00]"
  },
  {
    "0": "[code: 500]"
  }
]
```

Позиция в тексте

Следующие символы позволяют указать положение шаблона относительно начала и конца строки, а также границ слов.

Представление	Описание
`^`	Начало текста (в многострочном режиме — начало строки).
`$`	Конец текста (в многострочном режиме — конец строки).
`\A`	Только в начале текста, в т. ч. в многострочном режиме.
`\z`	Только в конце текста, в т. ч. в многострочном режиме.
`\b`	Граница слова Unicode: символ, соответствующий классу `\w` с одной стороны, и символ, соответствующий классу `\W`, либо граница текста, с другой стороны.
`\B`	Не на границе слова Unicode.
`\b{start}`, `\<`	Начало слова Unicode: символ класса `\W` или начало текста слева, символ класса `\w` справа.
`\b{end}`, `\>`	Конец слова Unicode: символ класса `\w` слева, символ класса `\w` или конец текста справа.
`\b{start-half}`	Проверка начала слова Unicode только слева: символ класса `\W` или начало текста.
`\b{end-half}`	Проверка конца слова Unicode только справа: символ класса `\W` или конец текста.

Примеры

Следующая функция возвращает последовательности алфавитно-цифровых и соединительных символов на границе слова.

Пример 178. Исходный код

```
.res = parse_regex_all!("@guest, @superuser, @root", r'(?P<user>\b\w+)')
```

Пример 179. Результат

```
[
  {
    "user": "guest"
  },
```

```
  {
    "user": "superuser"
  },
  {
    "user": "root"
  }
]
```

Чтобы получить последовательности с символом `@`, изменим проверку начала слова.

Пример 180. Исходный код

```
.res = parse_regex_all!("@guest, @superuser, @root", r'(?P<user>\b{start-half}@\w+)')
```

Пример 181. Код

```
[
  {
    "user": "@guest"
  },
  {
    "user": "@superuser"
  },
  {
    "user": "@root"
  }
]
```

Флаги

Регулярные выражения позволяют использовать флаги — модификаторы, которые изменяют поведение поиска для всего выражения или группы. Флаги могут быть объединены: `r'(?ixm)pattern'`, `r'(?im)pattern'`. Флаги можно включать и выключать внутри выражения.

Опция	Описание	Состояние по умолчанию
`i`	Режим без учета регистра: буквы в выражении соответствуют и прописным, и строчным буквам в строке.	Выключен
`m`	Многострочный режим: • `^` соответствует началу новой строки. • `$` соответствует концу строки.	Выключен

Опция	Описание	Состояние по умолчанию
x	Режим с поддержкой комментариев. Строчные комментарии должны начинаться с символа #. Пробелы при включенном режиме игнорируются. Чтобы добавить в шаблон пробел, можно использовать экранирование: \ ` или шестнадцатеричный код: \x20.	Выключен
s	Разрешает метасимволу точки . соответствовать символу новой строки \n.	Выключен
U	Меняет местами значение квантификаторов x* и x*? (ленивый режим).	Выключен
u	Поддержка Unicode.	Включен

Примеры

Рассмотрим следующую функцию:

Пример 182. Исходный код

```
.res = parse_regex!("AaAaAbbBBBb", r'(?i)a+(?-i)b+', numeric_groups: true)
```

Пример 183. Результат

```
{
  "0": "AaAaAbb"
}
```

В начале регулярного выражения включается регистронезависимый режим, поэтому a+ соответствует буквам a и A. Однако b+, после выключения регистронезависимого режима, соответствует только строчной b.

В следующем примере используется захват именованных групп, чтобы изменить формат даты.

Пример 184. Исходный код

```
replace_with("2022-01-05, 2023-08-25 и 2024-10-18", r'(?x)
  (?P<y>\d{4}) # год из 4 цифр Unicode
  -
  (?P<m>\d{2}) # месяц из 2 цифр Unicode
  -
  (?P<d>\d{2}) # день из 2 цифр Unicode
') -> |match| {
    to_string(match.d) + "/" + to_string(match.m) + "/" + to_string(match.y)
}
```

Пример 185. Результат

```
"05/01/2022, 25/08/2023 и 18/10/2024"
```

Массив

Литерал *массива* представляет собой набор выражений, разделенных запятыми, который создает непрерывный массив. Массив может быть динамически расширен.

Примеры

Пример 186. Пустой массив

```
[]
```

Пример 187. Массив из строк

```
["первый", "второй", "третий"]
```

Пример 188. Массив из литералов разного типа, включая объект

```
["смешанный", 1, 1.0, true, false, {"foo": "bar"}]
```

Пример 189. Вложенные массивы

```
[
  "первый-уровень",
  [
    "второй-уровень",
    [
      "третий-уровень"
    ]
  ]
]
```

Пример 190. Массив из динамических выражений

```
[.поле1, .поле2, to_int!("2"), variable_1]
```

Пример 191. Массив из динамических выражений

```
[
  "выражения",
  1 + 2,
  2 == 5,
  true || false
]
```

Объект

Литерал *объекта* представляет собой расширяемую структуру ключ/значение, которая синтаксически эквивалентна JSON-объекту.

Верно сформированный JSON-документ является допустимым объектом VRL.

Характеристики

Упорядочивание

Поля объекта упорядочены по ключу в алфавитном порядке по возрастанию. Таким образом, операции, такие как кодирование в JSON, производят строку с ключами, упорядоченными в алфавитном порядке.

Примеры

Пример 192. Объект с вложенными массивом и объектом

```
{
  "поле1": "значение1",
  "поле2": [
      "значение2",
      "значение3",
      "значение4"
   ],
  "поле3": {
    "поле4": "значение5"
  }
}
```

Пример 193. Массив со значениями, представленными путями и переменными

```
{
  "поле1": ."некий_путь",
  "поле2": some_variable,
  "поле3": {
    "вложенное_поле": "некое значение"
  }
}
```

Динамические выражения

Переменная

Выражение *переменной* именует переменные. Название переменной представляет собой последовательность из одной или нескольких букв, цифр и подчеркиваний. Первый символ в названии переменной должен быть буквой или цифрой.

Рекомендуется использовать осмысленные названия переменных.

Синтаксис

```
first ~ (trailing)*
```

Элемент	О писание
first	Первый символ может быть только буквенно-цифровым символом (`a–z`, `A–Z`, `0–9`).
trailing	Последующие символы должны содержать только буквенно-цифровые символы ASCII и подчеркивания (`a–z`, `A–Z`, `0–9`, `_`).

Примеры

Простая переменная

Пример 194. Программа VRL

```
my_variable = 1
```

Пример 195. Результат

```
1
```

Переменная с путем

Пример 196. Программа VRL

```
my_object = { "one": 1 }
my_object.one
```

Пример 197. Результат

```
1
```

Элемент	Описание
target	Цель должна быть путем с возможной второй переменной для обработки ошибок, если правая часть может возвращать ошибки.
error	Позволяет необязательное присваивание ошибок, когда выражение справа может возвращать ошибки. Это обычно используется при вызове функций, которые могут возвращать ошибки.
operator	Оператор разделяет `target` и `expression` и определяет условия присваивания.

Элемент	Описание
expression	Если `target` — это переменная, `expression` может быть любым выражением. Если `target` — это путь, `expression` может быть любым выражением, которое возвращает поддерживаемый тип значения объекта (то есть выражение, не являющееся регулярным).

Примеры

Присваивание пути

Пример 198. Программа VRL

```
.message = "Hello, World!"
```

Присваивание по вложенному пути

Пример 199. Программа VRL

```
.parent.child = "Hello, World!"
```

Двойное присваивание

Пример 200. Программа VRL

```
.first = .second = "Hello, World!"
```

Присваивание элемента массива

Пример 201. Программа VRL

```
.array[1] = "Hello, World!"
```

Присваивание переменной

Пример 202. Программа VRL

```
my_variable = "Hello, World!"
```

Пример 203. Результат

```
Hello, World!
```

Присваивание со слиянием объектов

Пример 204. Программа VRL

```
my_variable = {"message": "Hello, World!"}
```

```
my_variable |= {"level": "info"}
```

Пример 205. Результат

```
{
  "level": "info",
  "message": "Hello, World!"
}
```

Присваивание с возможностью ошибки (успех)

Пример 206. Программа VRL

```
.parsed, .err = parse_json("{\"Hello\": \"World!\"}")
```

Присваивание с возможностью ошибки (ошибка)

Пример 207. Программа VRL

```
.parsed, .err = parse_json("malformed")
```

Путь

Выражение *пути* представляет собой последовательность сегментов, разделенных точками, которые определяют местоположение значения внутри объекта. Ведущая "." означает, что путь указывает на событие. Ведущий "%" означает, что путь указывает на *метаданные* события.

Синтаксис

```
root_path ~ path_segments
```

Элемент	Описание
root_path	Все пути должны начинаться с `.` или `%`: • Символ `.` представляет корень события. • Символ `%` представляет корень метаданных события.
path_segments	Сегменты пути обозначают сегмент вложенного пути. Каждый сегмент должен быть отделен точкой в начале и может содержать только буквенно-цифровые символы и подчеркивание (`a–z`, `A–Z`, `0–9`, `_`). Сегменты, содержащие символы вне этого диапазона, должны быть заключены в кавычки.

Примеры

Корневой путь события

Пример 208. Событие Vector

```
{
  "message": "Hello, World!"
}
```

Пример 209. Программа VRL

```
.
```

Пример 210. Результат

```
{
  "message": "Hello, World!"
}
```

Корневой путь метаданных

Пример 211. Событие Vector

```
{
  "message": "Hello, World!"
}
```

Пример 212. Программа VRL

```
%
```

Пример 213. Результат

```
{}
```

Путь верхнего уровня

Пример 214. Событие Vector

```
{
  "message": "Hello, World!"
}
```

Пример 215. Программа VRL

```
.message
```

Пример 216. Результат

```
Hello, World!
```

Вложенный путь

Пример 217. Событие Vector

```
{
  "parent": {
    "child": "Hello, World!"
  }
}
```

Пример 218. Программа VRL

```
.parent.child
```

Пример 219. Результат

```
Hello, World!
```

Объединение вложенных путей

Пример 220. Событие Vector

```
{
  "grand_parent": {
    "parent2": {
      "child": "Hello, World!"
    }
  }
}
```

Пример 221. Программа VRL

```
.grand_parent.parent1.child || .grand_parent.parent2.child
```

Пример 222. Результат

```
Hello, World!
```

Путь в кавычках

Пример 223. Событие Vector

```
{
```

```
  "parent.key.with.special characters": {
    "child": "Hello, World!"
  }
}
```

Пример 224. Программа VRL

```
."parent.key.with.special characters".child
```

Пример 225. Результат

```
Hello, World!
```

Путь к первому элементу массива

Пример 226. Событие Vector

```
{
"массив": [
  "первый",
  "второй"
]
}
```

Пример 227. Программа VRL

```
."массив"[0]
```

Пример 228. Результат

```
"первый"
```

Путь ко второму элементу массива

Пример 229. Событие Vector

```
{
  "массив": [
    "первый",
    "второй"
  ]
}
```

Пример 230. Программа VRL

```
."массив"[1]
```

Пример 231. Результат

```
"второй"
```

Индекс

Выражение *индекса* обозначает элемент массива. Индексы массивов в VRL начинаются с нуля.

Синтаксис

```
"[" ~ index ~ "]"
```

Элемент	Описание
index	Индекс представляет собой нулевую позицию элемента.

Примеры

Выражение индекса массива

Пример 232. Событие Vector

```
{
  "массив": [
    "первый",
    "второй"
  ]
}
```

Пример 233. Программа VRL

```
."массив"[0]
```

Пример 234. Результат

```
"первый"
```

Присваивание

Выражение *присваивания* присваивает результат выражения справа от знака равенства цели слева (пути или переменной).

Чтобы избежать лишних вычислений, старайтесь минимизировать копирование исходного объекта.

Синтаксис

```
target ~ ("," ~ error)? ~ operator ~ expression
```

Аргумент	Значение
target	Цель должна быть переменной, индексом или путем. Возможно присваивание сразу нескольким целям, разделенным оператором `=`.
error	Позволяет необязательное присваивание ошибок, когда выражение справа может возвращать ошибки. Это обычно используется при вызове функций, которые могут возвращать ошибки.
operator	Оператор разделяет `target` и `expression` и определяет условия присваивания: • `=` — обычное присваивание; • `|=` — присваивание со слиянием объектов.
expression	Если `target` — это переменная, `expression` может быть любым выражением. Если `target` — это путь, `expression` может быть любым выражением, которое возвращает поддерживаемый тип значения объекта (то есть, выражение, не являющееся регулярным).

Характеристики

Двойное присваивание

Оператор `=` может использоваться для присваивания нескольким целям сразу. Двойное присваивание неприменимо, если используется присваивание ошибок.

Присваивание со слиянием

Оператор `|=` обновляет поля в целевом объекте на основе содержимого объекта из `expression`. При этом:

- Поля, которые есть только в целевом объекте, остаются неизменными.
- Поля, которые есть только во втором объекте, добавляются в целевой.
- Если поле существует в обоих объектах, значение поля из `expression` заменяет значение из целевого объекта, вложенные структуры при этом не учитываются.

Чтобы при слиянии объединить вложенные поля, используйте функцию `merge` с параметром `deep=true`.

Примеры

Присваивание пути

Пример 235. Программа VRL

```
.message = "Hello, World!"
```

Пример 236. Результат

```
{
  "message": "Hello, World!",
}
```

Присваивание по вложенному пути

Пример 237. Программа VRL

```
.parent.child = "Hello, World!"
```

Двойное присваивание

Пример 238. Программа VRL

```
.first = .second = "Hello, World!"
```

Пример 239. Результат

```
{
  "first": "Hello, World!",
  "second": "Hello, World!"
}
```

Присваивание элементу массива

Пример 240. Программа VRL

```
.array[1] = "Hello, World!"
```

Присваивание переменной

Пример 241. Программа VRL

```
my_variable = "Hello, World!"
```

Пример 242. Результат

```
Hello, World!
```

Присваивание со слиянием объектов

Пример 243. Программа VRL

```
my_variable = {
  "parent1": {
    "child1": 1,
```

```
    "child2": 2
  },
  "parent2": {
    "child3": 3
  }
}
my_variable |= {
  "parent1": {
    "child2": 4,
    "child5": 5
  }
}
```

Пример 244. Результат

```
{
  "parent1": {
    "child2": 4,
    "child5": 5
  },
  "parent2": {
    "child3": 3
  }
}
```

Присваивание с возможностью ошибки (успех)

Пример 245. Программа VRL

```
.parsed, .err = parse_json("{\"Hello\": \"World!\"}")
```

Пример 246. Результат

```
{
  "parsed": {
    "Hello": "World!"
  }
}
```

Присваивание с возможностью ошибки (ошибка)

Пример 247. Программа VRL

```
.parsed, .err = parse_json("malformed")
```

Пример 248. Результат

```
{
```

```
  "err": "malformed"
}
```

Арифметическое выражение

Арифметическое выражение выполняет операцию над двумя выражениями (операндами) согласно оператору.

Хотя арифметическое выражение обычно применяется к числам, его можно и с другими типами, такими как строки.

Синтаксис

```
expression ~ (operator ~ expression)+
```

Элемент	Описание
expression	Выражение может быть любым выражением, которое возвращает тип, допустимый для оператора.
operator	Оператор определяет операцию, выполняемую над операндами слева и справа. Существующие операторы: • `+` — сложение чисел или конкатенация строк; • `-` — вычитание чисел; • `*` — перемножение чисел или повторение строки указанное число раз; • `/` — деление чисел (не целочисленное).

Характеристики

Арифметические операции над числами разных типов возвращают числа с плавающей точкой.

Действует стандартный приоритет операций: умножение и деление выполняются до сложения и вычитания. Для изменения приоритета используются скобки.

Примеры

Сложение целых чисел

Пример 249. Программа VRL

```
1 + 1
```

Пример 250. Результат

```
2
```

Сложение чисел с плавающей точкой

Пример 251. Программа VRL

```
0.1 + 0.2
```

Пример 252. Результат

```
0.30000000000000004
```

Некоторые десятичные дроби не могут быть точно представлены в двоичном виде, что приводит к небольшим неточностям в вычислениях (например, `0.1 + 0.2 != 0.3`).

Сложение чисел разных типов

Пример 253. Программа VRL

```
1 + 1.0
```

Пример 254. Результат

```
2
```

Сложение строк

Пример 255. Программа VRL

```
"Hello" + ", " + "World!"
```

Пример 256. Результат

```
Hello, World!
```

Вычитание целых чисел

Пример 257. Программа VRL

```
2 - 1
```

Пример 258. Результат

```
1
```

Вычитание чисел с плавающей точкой

Пример 259. Программа VRL

```
2.0 - 1.0
```

Пример 260. Результат

```
1
```

Вычитание чисел разного типа

Пример 261. Программа VRL

```
2.0 - 1
```

Пример 262. Результат

```
1
```

Перемножение целых чисел

Пример 263. Программа VRL

```
2 * 1
```

Пример 264. Результат

```
2
```

Перемножение чисел с плавающей точкой

Пример 265. Программа VRL

```
2.0 * 1.0
```

Пример 266. Результат

```
2
```

Перемножение чисел разного типа

Пример 267. Программа VRL

```
2.0 * 1
```

Пример 268. Результат

```
2
```

Умножение строки на число

Пример 269. Программа VRL

```
"строка" * 2
```

Пример 270. Результат

```
"строкастрока"
```

Деление целых чисел

Пример 271. Программа VRL

```
2 / 1
```

Пример 272. Результат

```
2
```

Деление чисел с плавающей точкой

Пример 273. Программа VRL

```
2.0 / 1.0
```

Пример 274. Результат

```
2
```

Деление чисел разных типов

Пример 275. Программа VRL

```
2.0 / 1
```

Пример 276. Результат

```
2
```

Стандартный приоритет операций

Пример 277. Программа VRL

```
5 + 6 x 9 - 7
```

Пример 278. Результат

```
52
```

Задание приоритета с помощью скобок

Пример 279. Программа VRL

```
(5 + 6) x (9 - 7)
```

Пример 280. Результат

```
22
```

Для нахождения остатка используйте функцию `mod`.

Пример 281. Программа VRL

```
mod(3, 2)
```

Пример 282. Результат

```
1
```

Сравнение

Выражение *сравнения* сравнивает два выражения (операнда) и возвращает логическое значение в соответствии с оператором.

Синтаксис

```
expression ~ operator ~ expression
```

Элемент	Описание
expression	Операнд может быть любым выражением, которое возвращает тип, допустимый для оператора.

Элемент	Описание
operator	Оператор определяет операцию, выполняемую над операндами слева и справа: • == — равно; • != — неравно; • > — больше; • >= — больше или равно; • < — меньше; • <= — меньше или равно.

Характеристики

Равенство и неравенство Операторы == и != предназначены для сравнения между собой любых значений. Числовые значения разных типов могут быть равными, другие значения считаются равными, только если относятся к одному типу.

Сравнение порядка Операторы сравнения >, <, >= и <= применимы только к числам, строкам и меткам времени.

При сравнении строк используется лексикографический порядок, где символы сравниваются последовательно, на основе их номеров в таблице Unicode.

При сравнении меток времени большими считаются метки более поздней даты.

Примеры

Равенство чисел

Пример 283. Программа VRL

```
1 == 1.0
```

Пример 284. Результат

```
true
```

Неравенство чисел

Пример 285. Программа VRL

```
2 * 2 != 5
```

Пример 286. Результат

```
true
```

Больше или равно

Пример 287. Программа VRL

```
2 >= 2.0
```

Пример 288. Результат

```
true
```

Больше

Пример 289. Программа VRL

```
2 > 1
```

Пример 290. Результат

```
true
```

Меньше или равно

Пример 291. Программа VRL

```
2.0 <= 2
```

Пример 292. Результат

```
true
```

Меньше

Пример 293. Программа VRL

```
1 < 2
```

Пример 294. Результат

```
true
```

Сравнение временных меток

Пример 295. Программа VRL

```
t'2024-04-04T22:22:22.234142+01:00' > t'2024-04-04T22:22:22.234142+04:00'
```

Пример 296. Результат

```
true
```

Сравнение строк

Пример 297. Программа VRL

```
"ё" > "я"
```

Пример 298. Результат

```
true
```

Сравнение объектов

Пример 299. Программа VRL

```
{"key1": "value1", "key2": "value2"} == {"key2": "value2", "key1": "value1"}
```

Пример 300. Результат

```
true
```

Сравнение пустых значений

Пример 301. Программа VRL

```
null == null
```

Пример 302. Результат

```
true
```

Сравнение значений разных типов

Пример 303. Программа VRL

```
2 == "2"
```

Пример 304. Результат

```
false
```

Логическое выражение

Выражение *логического* типа сравнивает два выражения (операнды), прекращая вычисление на последнем выражении, вычисленном в соответствии с оператором.

При использовании нескольких логических операторов важно явно задавать приоритет операций с помощью скобок, чтобы избежать двусмысленных условий.

Синтаксис

Пример 305. Унарные операции

```
operator ~ expression
```

Пример 306. Бинарные операции

```
expression ~ operator ~ expression
```

Элемент	Описание
expression	Операнд может быть любым выражением, которое возвращает допустимый тип, определенный оператором.
operator	Оператор определяет операцию, выполняемую над операндами: • `&&` — конъюнкция (AND); • `||` — дизъюнкция (OR); • `!` — отрицание (NOT).

Примеры

AND

Пример 307. Программа VRL

```
true && true
```

Пример 308. Результат

```
true
```

OR с логическим значением

Пример 309. Программа VRL

```
false || "foo"
```

Пример 310. Результат

```
foo
```

OR с пустым значением

Пример 311. Программа VRL

```
null || "foo"
```

Пример 312. Результат

```
foo
```

NOT

Пример 313. Программа VRL

```
!false
```

Пример 314. Результат

```
true
```

Вызов функции

Выражение *вызова функции* вызывает встроенные функции VRL.

Синтаксис

```
function ~ abort? ~ "(" ~ arguments? ~ ")" ~ closure?
```

Элемент синтаксиса	**Значение**
function	Имя встроенной функции.

Элемент синтаксиса	Значение

abort	Прерывание — литерал `!`, который можно использовать с функциями, возвращающими ошибки, для прерывания программы, если функция возвращает ошибку:
	
	`result = f!()`
	
	В противном случае ошибки должны обрабатываться:
	
	`result, err = f()`
	
	Отсутствие обработки ошибок от функций, которые могут возвращать ошибку, приводит к ошибкам на этапе компиляции. Смотрите Справочник ошибок VRL для дополнительной информации.

arguments	Выражения, разделенные запятыми, которые могут дополнительно предваряться документированным именем.

closure	Замыкание — необязательный фрагмент кода, разрешенный вызовом функции. Замыкание применяется преимущественно в функциях, выполняющих итерации по коллекциям. Пример синтаксиса:
	
	`for_each([]) -> |index, value| {`
	`...`
	`}`

Примеры

Позиционный вызов функции

Пример 315. Программа VRL

```
split("hello, world!", ", ")
```

Пример 316. Результат

```
[
"hello",
"world!"
]
```

Вызов функции с именованными аргументами (упорядоченными)

Пример 317. Программа VRL

```
split("hello, world!", pattern: ", ")
```

Пример 318. Результат

```
[
  "hello",
  "world!"
]
```

Вызов функции с именованными аргументами (неупорядоченными)

Пример 319. Программа VRL

```
split(pattern: ", ", value: "hello, world!")
```

Пример 320. Результат

```
[
  "hello",
  "world!"
]
```

Объединение

Выражение *объединения* состоит из нескольких выражений (операндов), разделенных оператором объединения ??. Логика объединения позволяет выбирать из списка выражений первое значение, которое соответствует заданному условию.

Оператор ?? полезен при обработке ошибок, так как позволяет задавать альтернативные способы обработки данных.

Синтаксис

```
expression ~ (operator ~ expression)+
```

Элемент	Описание
expression	Операнд может быть любым выражением.
operator	Оператор разделяет два или более выражения.

Примеры

Объединение при ошибках

Пример 321. Программа VRL

```
parse_syslog("не Syslog") ?? parse_common_log("не Common Log Format") ?? "malformed"
```

Пример 322. Результат

```
malformed
```

Условие

Выражение *условия (if)* определяет условное выполнение двух ветвей в зависимости от значения логического выражения. Если логическое выражение оценивается как истинное (`true`), выполняется ветвь "if". В противном случае выполняется ветвь "else", если она присутствует.

Синтаксис

```
"if" ~ predicate ~ block ~ ("else if" ~ predicate ~ block)* ~ ("else" ~ block)?
```

Элемент	Описание
predicate	Предикат *должен* быть выражением, которое приводится к значению логического типа. Если логическое значение не возвращается, возникает ошибка на этапе компиляции. Предикат может содержать несколько выражений. Несколько предикатов выражений должны быть заключены в скобки. Выражения должны быть разделены точкой с запятой (`;`) или новой строкой.

Примеры

Условие if true

Пример 323. Программа VRL

```
if true {
  "Hello, World!"
}
```

Пример 324. Результат

```
Hello, World!
```

Условие if false

Пример 325. Программа VRL

```
if false {
# not evaluated
  null
```

```
}
```

Пример 326. Результат

```
null
```

Условие if/else

Пример 327. Программа VRL

```
if false {
# not evaluated
  null
} else {
  "Hello, World!"
}
```

Пример 328. Результат

```
Hello, World!
```

Условие if … else if … else

Пример 329. Программа VRL

```
if false {
# not evaluated
  null
} else if false {
# not evaluated
  null
} else {
  "Hello, World!"
}
```

Пример 330. Результат

```
Hello, World!
```

Многострочное выражение

Пример 331. Программа VRL

```
x = 3
if (x = x + 1; x == 5) {
# not evaluated
  null
} else if (
```

```
  x = x + 1
  x == 5
) {
  "Hello, World!"
}
```

Пример 332. Результат

```
Hello, World!
```

Блок

Выражение *блока* представляет собой последовательность одного или нескольких выражений, заключенных в фигурные скобки.

Блоки не могут быть пустыми. Сочетание фигурных скобок без содержимого ({}) рассматривается как пустой объект.

Синтаксис

```
"{" ~ "NEWLINE"* ~ expressions ~ "NEWLINE"* ~ "}"
```

Элемент	Описание
function	Имя встроенной функции.
expressions	Одно или несколько выражений.

Примеры

Простой блок

Пример 333. Программа VRL

```
{
  message = "{\"Hello\": \"World!\"}"
  parse_json!(message)
}
```

Пример 334. Результат

```
{
  "Hello": "World!"
}
```

Присваивание в блоке

Пример 335. Программа VRL

```
.structured = {
  message = "{\"Hello\": \"World!\"}"
  parse_json!(message)
}
```

Прерывание

Выражение `abort` приводит к завершению программы VRL, прерывая любые изменения, вносимые в событие.

Синтаксис

```
abort ~ message?
```

Элемент	Описание
message	Необязательное отладочное сообщение, которое может быть использовано в диагностических целях и включается в метаданные события, отброшенного при преобразовании `remap`.

Примеры

Игнорирование недопустимых событий

Пример 336. Событие Vector

```
{
  "message": "hello world"
}
```

Пример 337. Программа VRL

```
if contains(string!(.message), "hello") {
  abort
}
.message = "not hello world"
```

Пример 338. Результат

```
{
  message": "hello world"
}
```

Функции

В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Если нужно неоднократно обращаться к блоку кода в правилах нормализации и декларативных правилах корреляции, можно создать глобальную функцию.

Синтаксис

Синтаксис вызова функций VRL подробно описан в статье Вызов функции. Рассмотрим следующую спецификацию функции:

```
redact(value: <строка | объект | массив>, filters: <массив>, [redactor: <строка |
объект> ]))
:: <строка | объект | массив>
```

Описанную функцию можно вызвать так:

Пример 339. Исходный код

```
redact("Userid 123456-7890", filters: [r'\d+\-\d+'], redactor: "sha2")
```

Возможность функции возвращать ошибки

Некоторые функции VRL могут быть помечены как возвращающие ошибки или не возвращающие ошибки. Когда функция определена как возвращающая ошибки, она может вернуть ошибку во время выполнения, требуя обработки ошибки перед компиляцией программы.

Если функция определена как не возвращающая ошибки, это означает, что **при правильных аргументах функции** функция никогда не может завершиться с ошибкой во время выполнения, и поэтому обработка ошибки не требуется.

Важно помнить, что даже если функция не должна возвращать ошибку, но при этом один из ее аргументов возвращает ошибку во время выполнения, вся функция будет считаться ошибочной. В такой ситуации необходимо обработать ошибку.

Следует отметить, что даже если функция определена как **не возвращающая ошибки**, но при этом любой из ее аргументов возвращает ошибку во время выполнения, функция рассматривается как **возвращающая ошибки**, и, следовательно, в этом случае необходимо будет обработать ошибку.

Компилятор VRL обеспечивает обработку всех потенциальных ошибок в программе, что исключает

сбои во время выполнения.

Функции разбора

В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции разбора согласно регулярному выражению

parse_regex

Разбирает строку `value` с использованием предоставленного регулярного выражения `pattern`. Эта функция отличается от функции `parse_regex_all` тем, что ищет только первое совпадение.

Функция возвращает объект с полями, соответствующими именованным группам. При включенной опции `numeric_groups` объект также включает поля, ключи которых являются последовательными номерами:

- поле `"0"` содержит совпадение полностью;
- последующие поля содержат захваченные группы по порядку.

Спецификация функции

```
parse_regex(value: <строка>, pattern: <регулярное выражение>, [numeric_groups:
<логическое значение>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для поиска.		да
`pattern`	регулярное выражение	Регулярное выражение для поиска.		да
`numeric_groups`	логическое значение	Если `true`, будет возвращаться каждая захваченная группа в регулярном выражении. Индекс `0` будет содержать совпадение полностью.	`false`	нет

Примечания

Эта функция имеет особое поведение, о котором следует знать.

- Если в `pattern` нет именованных групп и значение `numeric_groups` равно `false`, функция

возвращает пустой объект.

- Все значения возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.

VRL стремится предложить специализированные функции для анализа распространенных форматов журналов. Поскольку регулярные выражения требуют больших ресурсов, рекомендуем использовать функцию `parse_regex`, только если нет специальной функции VRL для нужного формата.

Ошибки

Функция `parse_regex` **может возвращать ошибки**, для которых требуется обработка:

- Строку `value` не удалось разобрать с использованием предоставленного шаблона `pattern`.

Примеры

Разбор с использованием регулярного выражения (с захватом именованной группы)

Пример 340. Исходный код

```
parse_regex!("Connection attempt from 192.0.2.0, from 192.0.2.1.",
r'from (?P<ip>\d+\.\d+\.\d+\.\d+)')
```

Пример 341. Результат

```
{
  "ip": "192.0.2.0"
}
```

Разбор с использованием регулярного выражения (без захвата именованных групп)

Пример 342. Исходный код

```
parse_regex!("Connection attempt from 192.0.2.0, from 192.0.2.1.",
r'from (\d+\.\d+\.\d+\.\d+)',
numeric_groups: true)
```

Пример 343. Результат

```
{
  "0": "from 192.0.2.0",
  "1": "192.0.2.0"
}
```

parse_regex_all

Разбирает `value` с использованием предоставленного регулярного выражения `pattern`. Эта функция отличается от функции `parse_regex` тем, что ищет *все* совпадения, а не только первое.

Функция возвращает массив объектов с полями, соответствующими именованным группам. При включенной опции `numeric_groups` объекты также включают поля, ключи которых являются последовательными номерами:

- поле `"0"` содержит совпадение полностью;
- последующие поля содержат захваченные группы по порядку.

Спецификация функции

```
parse_regex_all(value: <строка>, pattern: <регулярное выражение>, [numeric_groups:
<логическое значение>])
:: <массив> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для поиска.		да
`pattern`	регулярное выражение	Регулярное выражение для поиска.		да
`numeric_groups`	логическое значение	Если `true`, будет возвращаться каждая захваченная группы в регулярном выражении. Индекс `0` будет содержать совпадение полностью.	`false`	нет

Примечания

Эта функция имеет особое поведение, о котором следует знать.

- VRL стремится предоставлять функции специального назначения для разбора общих форматов журналов. Прежде чем использовать функцию `parse_regex_all`, проверьте, существует ли уже функция VRL, которая разбирает нужный формат.
- Если в `pattern` нет именованных групп и значение `numeric_groups` равно `false`, функция возвращает массив пустых объектов.
- Все значения возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.

Ошибки

Функция `parse_regex_all` **может возвращать ошибки**, для которых требуется обработка:

- `value` не удалось разобрать с использованием предоставленного шаблона `pattern`.

Примеры

Разбор с использованием регулярного выражения (все совпадения)

Пример 344. Исходный код

```
parse_regex_all!("Connection attempt from 192.0.2.0, from 192.0.2.1.",
```

```
r'from (?P<ip>\d+\.\d+\.\d+\.\d+)',
numeric_groups: true)
```

Пример 345. Результат

```
[
  {
    "0": "from 192.0.2.0",
    "1": "192.0.2.0",
    "ip": "192.0.2.0"
  },
  {
    "0": "from 192.0.2.1",
    "1": "192.0.2.1",
    "ip": "192.0.2.1"
  }
]
```

Функции разбора форматов логов

parse_apache_log

Разбирает строки журнала доступа и ошибок Apache. Строки могут иметь формат `common`, `combined` или `error`.

Спецификация функции

```
parse_apache_log(value: <строка>, format: <строка>, [timestamp_format: <строка>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`format`	строка	Формат, используемый для разбора журнала.		да
`timestamp_format`	строка	Формат даты/времени, используемый для кодирования метки времени. Время разбирается в локальном часовом поясе, если метка времени не указывает часовой пояс.	`%d/%b/%Y:%T %z`	нет

Примечания

У этой функции есть особое поведение, о котором следует знать:

- Отсутствие информации в сообщении журнала может быть обозначено `-`. Эти поля опущены в результате.

Ошибки

Функция `parse_apache_log` **может возвращать ошибки**, для которых требуется обработка:

- `value` не соответствует указанному формату.
- `timestamp_format` не является допустимой строкой формата.
- Метку времени в `value` не удалось разобрать с использованием предоставленного `timestamp_format`.

Примеры

Разбор строки в формате журнала Apache (common)

Пример 346. Исходный код

```
parse_apache_log!("192.0.2.0 bob frank [10/Oct/2000:13:55:36 -0700] \"GET
/apache_pb.gif HTTP/1.0\" 200 2326",
format: "common")
```

Пример 347. Результат

```
{
  "host": "192.0.2.0",
  "identity": "bob",
  "message": "GET /apache_pb.gif HTTP/1.0",
  "method": "GET",
  "path": "/apache_pb.gif",
  "protocol": "HTTP/1.0",
  "size": 2326,
  "status": 200,
  "timestamp": "2000-10-10T20:55:36Z",
  "user": "frank"
}
```

Разбор строки в формате журнала Apache (combined)

Пример 348. Исходный код

```
parse_apache_log!(
s'192.0.2.0 bob frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200
2326 "http://www.example.com/vertical/channels/front-end/bandwidth" "Mozilla/5.0 (X11;
Linux i686; rv:5.0) Gecko/1945-10-12 Firefox/37.0"',
"combined")
```

Пример 349. Результат

```
{
  "agent": "Mozilla/5.0 (X11; Linux i686; rv:5.0) Gecko/1945-10-12 Firefox/37.0",
  "host": "192.0.2.0",
```

```
  "identity": "bob",
  "message": "GET /apache_pb.gif HTTP/1.0",
  "method": "GET",
  "path": "/apache_pb.gif",
  "protocol": "HTTP/1.0",
  "referrer": "http://www.example.com/vertical/channels/front-end/bandwidth",
  "size": 2326,
  "status": 200,
  "timestamp": "2000-10-10T20:55:36Z",
  "user": "frank"
}
```

Разбор строки в формате журнала Apache (ошибка)

Пример 350. Исходный код

```
parse_apache_log!(
s'[01/Mar/2021:12:00:19 +0000] [ab:alert] [pid 4803:tid 3814] [client
147.159.108.175:24259] I will bypass the haptic COM bandwidth, that should matrix the
CSS driver!',
"error")
```

Пример 351. Результат

```
{
  "client": "147.159.108.175",
  "message": "I will bypass the haptic COM bandwidth, that should matrix the CSS
driver!",
  "module": "ab",
  "pid": 4803,
  "port": 24259,
  "severity": "alert",
  "thread": "3814",
  "timestamp": "2021-03-01T12:00:19Z"
}
```

rv_parse_avro

Разбирает `value` как файл формата Apache Avro, который содержит метаданные со схемой. Если сообщение не содержит схемы, используйте функцию `rv_parse_avro_raw`.

Спецификация функции

```
rv_parse_avro(value: <строка>, [reader_schema: <строка>])
:: <массив объектов>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление сообщения Avro		да
`reader_schema`	JSON	Схема для считывания в формате JSON. Строка должна быть представлена литералом, без использования переменных.		нет

Примечания

Особенности функции, о которых нужно знать:

- Использование схемы считывания (параметр `reader_schema`) может замедлить выполнение функции.
- Тип `Decimal` в сообщениях Avro не поддерживается.

Ошибки

Функция `rv_parse_avro` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректным сообщением Avro;
- `value` закодировано не в UTF-8;
- `reader_schema` не является строкой;
- невозможно разобрать предоставленную `reader_schema`;
- значения типов `TimeMicros`, `TimeMillis`, `Float` и `Double` в разобранном сообщении не являются числовыми.

Примеры

Базовый разбор

Пример 352. Исходный код

```
# .message -- путь к сообщению в формате Avro
rv_parse_avro!(.message)
```

Пример 353. Результат

```
# пример массива объектов
[
  {
    "entity": "001",
    "time": "1433269388"
  },
  {
    "entity": "002",
    "time": "1433270488"
```

```
  },
]
```

С указанием схемы

Пример 354. Исходный код

```
# .message -- путь к сообщению в формате Avro
rv_parse_avro!(.message,
reader_schema: s'{"type": "record", "fields": [{"name": "element", "type": "string"},
{"name": "time", "type": "string"}]}')
```

Пример 355. Результат

```
# пример массива объектов
[
  {
    "element": "001",
    "time": "1433269388"
  },
  {
    "element": "002",
    "time": "1433270488"
  },
]
```

rv_parse_avro_raw

Разбирает `value` как отдельную запись Apache Avro, сериализованную без заголовков или встроенной схемы. Функция требует предоставлять схему расшифровки данных.

Спецификация функции

```
rv_parse_avro_raw(value: <строка>, decoder_schema: <avro_schema>, [reader_schema:
<строка>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление сообщения Avro		да
`decoder_schema`	строка	Схема для расшифровки данных. Строка должна быть представлена литералом, без использования переменных.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`reader_schema`	JSON	Схема в формате JSON для валидации расшифрованных данных. Строка должна быть представлена литералом, без использования переменных.		нет

Примечания

Особенности функции, о которых нужно знать:

- Использование схемы считывания (параметр `reader_schema`) может замедлить выполнение функции.
- Тип `Decimal` в сообщениях Avro не поддерживается.

Ошибки

Функция `rv_parse_avro_raw` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректным сообщением Avro;
- `decoder_schema` не является строкой;
- `reader_schema` не является строкой;
- невозможно разобрать предоставленную `decoder_schema`;
- невозможно разобрать предоставленную `reader_schema`;
- значения типов `TimeMicros`, `TimeMillis`, `Float` и `Double` в разобранном сообщении не являются числовыми.

Примеры

Базовый разбор

Пример 356. Исходный код

```
# .message -- путь к сообщению в формате Avro
rv_parse_avro_raw!(.message,
decoder_schema: s'{"type": "record", "fields": [{"name": "element", "type": "string"},
{"name": "time", "type": "string"}]}')
```

Пример 357. Результат

```
# пример объекта
{
  "element": "value",
  "time": "1433270488"
}
```

С указанием схемы считывания

Пример 358. Исходный код

```
# .message -- путь к сообщению в формате Avro
rv_parse_avro_raw!(.message,
decoder_schema: s'{"type": "record", "fields": [{"name": "element", "type": "string"},
{"name": "time", "type": "string"}]}',
reader_schema: s'{"type": "record", "fields": [{"name": "element", "type":
"string"}]}')
```

Пример 359. Результат

```
# пример объекта
{
  "element": "value"
}
```

parse_aws_alb_log

Разбирает `value` в формате журнала доступа Application Load Balancer в AWS (Amazon Web Services).

Спецификация функции

```
parse_aws_alb_log(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Журнал доступа Application Load Balancer.		да

Ошибки

Функция `parse_aws_alb_log` **может возвращать ошибки**, для которых требуется обработка:

- `value` не соответствует указанному формату лога AWS ALB.

Примеры

Разбор лога AWS ALB

Пример 360. Исходный код

```
parse_aws_alb_log!(
"http 2018-11-30T22:23:00.186641Z app/my-loadbalancer/50dc6c495c0c9188
192.168.131.39:2817 - 0.000 0.001 0.000 200 200 34 366 \"GET
http://www.example.com:80/ HTTP/1.1\" \"curl/7.46.0\" - -
arn:aws:elasticloadbalancing:us-east-2:123456789012:targetgroup/my-
targets/73e2d6bc24d8a067 \"Root=1-58337364-23a8c76965a2ef7629b185e3\" \"-\" \"-\" 0
```

```
2018-11-30T22:22:48.364000Z \"forward\" \"-\" \"-\" \"-\" \"-\" \"-\" \"-\""
)
```

Пример 361. Результат

```
{
  "actions_executed": "forward",
  "chosen_cert_arn": null,
  "classification": null,
  "classification_reason": null,
  "client_host": "192.168.131.39:2817",
  "domain_name": null,
  "elb": "app/my-loadbalancer/50dc6c495c0c9188",
  "elb_status_code": "200",
  "error_reason": null,
  "matched_rule_priority": "0",
  "received_bytes": 34,
  "redirect_url": null,
  "request_creation_time": "2018-11-30T22:22:48.364000Z",
  "request_method": "GET",
  "request_processing_time": 0,
  "request_protocol": "HTTP/1.1",
  "request_url": "http://www.example.com:80/",
  "response_processing_time": 0,
  "sent_bytes": 366,
  "ssl_cipher": null,
  "ssl_protocol": null,
  "target_group_arn": "arn:aws:elasticloadbalancing:us-east-
2:123456789012:targetgroup/my-targets/73e2d6bc24d8a067",
  "target_host": null,
  "target_port_list": [],
  "target_processing_time": 0.001,
  "target_status_code": "200",
  "target_status_code_list": [],
  "timestamp": "2018-11-30T22:23:00.186641Z",
  "trace_id": "Root=1-58337364-23a8c76965a2ef7629b185e3",
  "traceability_id": null,
  "type": "http",
  "user_agent": "curl/7.46.0"
}
```

parse_aws_cloudwatch_log_subscription_message

Разбирает события журналов AWS CloudWatch, настроенных через подписки Cloudwatch, из источника `aws_kinesis_firehose`.

Спецификация функции

```
parse_aws_cloudwatch_log_subscription_message(value: <строка>)
```

```
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление сообщения для разбора.		да

Ошибки

Функция `parse_aws_cloudwatch_log_subscription_message` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является правильно отформатированным сообщением подписки AWS Cloudwatch Log.

Примеры

Разбор сообщения подписки AWS Cloudwatch Log

Пример 362. Исходный код

```
# .message -- пример пути к сообщению журнала AWS CloudWatch
parse_aws_cloudwatch_log_subscription_message!(.message)
```

Пример 363. Результат

```
{
  "log_events": [
    {
      "id": "35683658089614582423604394983260738922885519999578275840",
      "message": "{\"bytes\":26780,\"datetime\":\"14/Sep/2020:11:45:41
-0400\",\"host\":\"192.0.2.1\",\"method\":\"PUT\",\"protocol\":\"HTTP/1.0\",\"referer\
":\"https://www.principalcross-
platform.io/markets/ubiquitous\",\"request\":\"/expedite/convergence\",\"source_type\"
:\"stdin\",\"status\":301,\"user-identifier\":\"-\"}",
      "timestamp": "2020-09-14T19:09:29.039Z"
    }
  ],
  "log_group": "test",
  "log_stream": "test",
  "message_type": "DATA_MESSAGE",
  "owner": "111111111111",
  "subscription_filters": [
    "Destination"
  ]
}
```

parse_aws_vpc_flow_log

Разбирает `value` в формате журналов потокового контроля AWS Virtual Private Cloud (VPC Flow Logs).

Спецификация функции

```
parse_aws_vpc_flow_log(value: <строка>, [format: <строка>])
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Журнал потоков VPC.		да
`format`	строка	Формат журнала потоков VPC.		нет

Ошибки

Функция `parse_aws_vpc_flow_log` **может возвращать ошибки**, для которых требуется обработка:

- `value` имеет неправильный формат журнала потоков VPC AWS.

Примеры

Разбор журнала потоков VPC AWS (формат по умолчанию)

Пример 364. Исходный код

```
parse_aws_vpc_flow_log!("2 123456789010 eni-1235b8ca123456789 - - - - - - - 1431280876
1431280934 - NODATA")
```

Пример 365. Результат

```
{
  "account_id": "123456789010",
  "action": null,
  "bytes": null,
  "dstaddr": null,
  "dstport": null,
  "end": 1431280934,
  "interface_id": "eni-1235b8ca123456789",
  "log_status": "NODATA",
  "packets": null,
  "protocol": null,
  "srcaddr": null,
  "srcport": null,
  "start": 1431280876,
  "version": 2
}
```

Разбор журнала потоков VPC AWS (пользовательский формат)

Пример 366. Исходный код

```
parse_aws_vpc_flow_log!(
"- eni-1235b8ca123456789 10.0.1.5 10.0.0.220 10.0.1.5 203.0.113.5",
"instance_id interface_id srcaddr dstaddr pkt_srcaddr pkt_dstaddr"
)
```

Пример 367. Результат

```
{
  "dstaddr": "10.0.0.220",
  "instance_id": null,
  "interface_id": "eni-1235b8ca123456789",
  "pkt_dstaddr": "203.0.113.5",
  "pkt_srcaddr": "10.0.1.5",
  "srcaddr": "10.0.1.5"
}
```

parse_cef

Разбирает `value` в формате CEF (Common Event Format). Игнорирует все данные до заголовка CEF.

Рекомендуется использовать функцию `rv_parse_cef`, которая поддерживает заголовки syslog.

Спецификация функции

```
parse_cef(value: <строка>, [translate_custom_fields: <логическое значение>])
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора		да
`translate_custom_fields`	логическое значение	Преобразовывать пользовательские пары полей в формат `ключ: значение`	`false`	нет

Примечания

У этой функции есть специальное поведение, которое следует учитывать:

- Все значения возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.
- Обрамляющие кавычки удаляются из значений.
- Пустые значения возвращаются как пустые строки.

Ошибки

Функция `parse_cef` **может возвращать ошибки**, для которых требуется обработка:

- `value` имеет неправильный формат строки CEF.

Примеры

Разбор сообщения, сгенерированного Privileged Threat Analytics (PTA)

Пример 368. Исходный код

```
parse_cef!(
"CEF:0|CyberArk|PTA|12.6|1|Suspected credentials theft|8|suser=mike2@prod1.example.com
shost=prod1.example.com src=198.51.100.0 duser=andy@dev1.example.com
dhost=dev1.example.com dst=203.0.113.0 cs1Label=ExtraData cs1=None cs2Label=EventID
cs2=52b06812ec3500ed864c461e deviceCustomDate1Label=detectionDate
deviceCustomDate1=1388577900000 cs3Label=PTAlink
cs3=https://198.51.100.0/incidents/52b06812ec3500ed864c461e cs4Label=ExternalLink
cs4=None"
)
```

Пример 369. Результат

```
{
  "cefVersion": "0",
  "cs1": "None",
  "cs1Label": "ExtraData",
  "cs2": "52b06812ec3500ed864c461e",
  "cs2Label": "EventID",
  "cs3": "https://198.51.100.0/incidents/52b06812ec3500ed864c461e",
  "cs3Label": "PTAlink",
  "cs4": "None",
  "cs4Label": "ExternalLink",
  "deviceCustomDate1": "1388577900000",
  "deviceCustomDate1Label": "detectionDate",
  "deviceEventClassId": "1",
  "deviceProduct": "PTA",
  "deviceVendor": "CyberArk",
  "deviceVersion": "12.6",
  "dhost": "dev1.example.com",
  "dst": "203.0.113.0",
  "duser": "andy@dev1.example.com",
  "name": "Suspected credentials theft",
  "severity": "8",
  "shost": "prod1.example.com",
  "src": "198.51.100.0",
  "suser": "mike2@prod1.example.com"
}
```

Сообщение с заголовком syslog, которое игнорируется

Пример 370. Исходный код

```
parse_cef!(
"Sep 29 08:26:10 host CEF:1|Security|threatmanager|1.0|100|worm successfully
stopped|10|src=198.51.100.0 dst=203.0.113.0 spt=1232"
)
```

Пример 371. Результат

```
{
  "cefVersion": "1",
  "deviceVendor": "Security",
  "deviceProduct": "threatmanager",
  "deviceVersion": "1.0",
  "deviceEventClassId": "100",
  "name": "worm successfully stopped",
  "severity": "10",
  "src": "198.51.100.0",
  "dst": "203.0.113.0",
  "spt": "1232"
}
```

Преобразование пользовательских полей

Пример 372. Исходный код

```
parse_cef!(
"CEF:0|Dev|firewall|2.2|1|Connection
denied|5|c6a1=2001:0DB8:2CA1:0000:0000:0567:5673:23b5 c6a1Label=Device IPv6 Address",
translate_custom_fields: true
)
```

Пример 373. Результат

```
{
  "cefVersion": "0",
  "deviceVendor": "Dev",
  "deviceProduct": "firewall",
  "deviceVersion": "2.2",
  "deviceEventClassId": "1",
  "name": "Connection denied",
  "severity": "5",
  "Device IPv6 Address": "2001:0DB8:2CA1:0000:0000:0567:5673:23b5"
}
```

rv_parse_cef

Разбирает `value` в формате Common Event Format (CEF). Поддерживает и обрабатывает необязательный заголовок syslog.

Спецификация функции

```
rv_parse_cef(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление записи CEF		да

Ошибки

Функция `rv_parse_cef` **может возвращать ошибки**, для которых требуется обработка:

- строка `value` не начинается с обязательного префикса `CEF:`;
- строка `value` не содержит обязательных полей и не является корректной записью CEF.

Примеры

Базовый разбор

Пример 374. Исходный код

```
rv_parse_cef!(s'CEF:0|Microsoft|Microsoft Windows||Microsoft-Windows-Security-
Auditing:4624|An account was successfully logged on.|Low| eventId=1013540004
externalId=4624')
```

Пример 375. Результат

```
{
  "deviceVendor": "Microsoft",
  "deviceProduct": "Microsoft Windows",
  "deviceVersion": "",
  "deviceEventClassId": "Microsoft-Windows-Security-Auditing:4624",
  "name": "An account was successfully logged on.",
  "severity": "Low",
  "eventId": "1013540004",
  "externalId": "4624"
}
```

Сообщение с заголовком syslog

Пример 376. Исходный код

```
value = r#"<134>1 2022-02-14T03:17:30-08:00 TEST
CEF:0|Vendor|Product|20.0.560|600|User Signed In|3|src=192.0.2.255 "#
rv_parse_cef!(value)
```

Пример 377. Результат

```
{
  ahost: "TEST",
  at: "2022-02-14T03:17:30-08:00",
  deviceProduct: "Product",
  deviceVendor: "Vendor",
  deviceVersion: "20.0.560",
  name: "User Signed In",
  severity: "3",
  signatureId: "600",
  src: "192.0.2.255",
  syslog_facility: "16",
  syslog_priority: "134",
  syslog_severity: "6",
})
```

parse_common_log

Разбирает `value` с использованием формата общего журнала Apache (Common Log Format, CLF).

Спецификация функции

```
parse_common_log(value: <строка>, [timestamp_format: <строка>])
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`timestamp_format`	строка	Формат даты и времени для кодирования метки времени.	`%d/%b/%Y:%T %z`	нет

Примечания

У этой функции есть специальное поведение, которое следует учитывать:

- Отсутствующая информация в журнальной записи может быть обозначена символом `-`. Эти поля опускаются в результате.

Ошибки

Функция `parse_common_log` **может возвращать ошибки**, для которых требуется обработка:

- `value` не соответствует формату общего журнала.
- `timestamp_format` не является допустимой строкой формата.
- Метку времени в `value` не удается разобрать с использованием предоставленного `timestamp_format`.

Примеры

Разбор сообщения в формате общего журнала (с форматом метки времени по умолчанию)

Пример 378. Исходный код

```
parse_common_log!("192.0.2.2 bob frank [10/Oct/2000:13:55:36 -0700] \"GET
/apache_pb.gif HTTP/1.0\" 200 2326")
```

Пример 379. Результат

```
{
  "host": "192.0.2.2",
  "identity": "bob",
  "user": "frank",
  "timestamp": "2000-10-10T20:55:36Z",
  "message": "GET /apache_pb.gif HTTP/1.0",
  "method": "GET",
  "path": "/apache_pb.gif",
  "protocol": "HTTP/1.0",
  "status": 200,
  "size": 2326
}
```

Разбор сообщения в формате общего журнала с пользовательским форматом метки времени

Пример 380. Исходный код

```
parse_common_log!(
"192.0.2.25 bob frank [2000-10-10T20:55:36Z] \"GET /apache_pb.gif HTTP/1.0\" 200
2326",
"%+"
)
```

Пример 381. Результат

```
{
  "host": "192.0.2.25",
  "identity": "bob",
  "user": "frank",
  "timestamp": "2000-10-10T20:55:36Z",
  "message": "GET /apache_pb.gif HTTP/1.0",
  "method": "GET",
  "path": "/apache_pb.gif",
  "protocol": "HTTP/1.0",
  "status": 200,
  "size": 2326
}
```

rv_extract_from_dit

Извлекает домен и имя пользователя из записи DIT (Data Information Tree), используемой в протоколе LDAP.

Спецификация функции

```
rv_extract_from_dit(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление записи DIT		да

Примечания

Особенность функции, о которой нужно знать:

- Результат возвращается в виде JSON-объекта с полями `"username"` и `"domain"`.
- Имя пользователя и домен возвращаются в нижнем регистре.

Ошибки

Функция `rv_extract_from_dit` **может возвращать ошибки**, для которых требуется обработка:

- `value` закодировано не в UTF-8.
- `value` не является корректной записью DIT.

Пример

Пример 382. Исходный код

```
rv_extract_from_dit!("CN=Dev-India,OU=Distribution
Groups,DC=gp,DC=gl,DC=example,DC=com")
```

Пример 383. Результат

```
{
  "domain": "gp.gl.example.com",
  "username": "dev-india"
}
```

parse_etld

Разбирает eTLD (effective top-level domain) из строки `value`, представляющей собой доменное имя.

Спецификация функции

```
parse_etld(value: <строка>, [plus_parts: <целое число>, psl: <строка>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка домена.		да
`plus_parts`	целое число	Число поддоменов, возвращаемых функцией.		нет
`psl`	строка	Путь к файлу или URL списка публичных суффиксов. По умолчанию используется список publicsuffix.org/list/public_suffix_list.dat.		нет

`parse_etld` возвращает структуру вида:

```
{
  "etld": "ac.uk",
  "etld_plus": "example.ac.uk",
  "known_suffix": true
}
```

- `etld` — имя домена;
- `etld_plus` — имя домена с поддоменами. Количество поддоменов определяется аргументом `plus_parts`;
- `known_suffix` — `true`, если суффикс входит в список, переданный в аргументе `psl`, `false` — иначе.

Ошибки

Функция `parse_etld` **может возвращать ошибки**, для которых требуется обработка:

- невозможно определить eTLD для строки `value`.

Примеры

Разбор eTLD

Пример 384. Исходный код

```
parse_etld!("sub.example.com")
```

Пример 385. Результат

```
{
  "etld": "com",
  "etld_plus": "com",
```

```
  "known_suffix": true
}
```

Разбор eTLD+1

Пример 386. Исходный код

```
parse_etld!("example.com", plus_parts: 1)
```

Пример 387. Результат

```
{
  "etld": "com",
  "etld_plus": "example.com",
  "known_suffix": true
}
```

Разбор eTLD с неизвестным суффиксом

Пример 388. Исходный код

```
parse_etld!("example.acmecorp")
```

Пример 389. Результат

```
{
  "etld": "acmecorp",
  "etld_plus": "acmecorp",
  "known_suffix": false
}
```

Разбор eTLD с другим PSL

Пример 390. Исходный код

```
parse_etld!("example.acmecorp",
# пример пути к списку доменов
psl: "resources/public_suffix_list.dat")
```

Пример 391. Результат

```
{
  "etld": "acmecorp",
  "etld_plus": "acmecorp",
  "known_suffix": false
}
```

dns_lookup

Выполняет DNS Lookup для доменного имени `value`.

Не рекомендуется применять `dns_lookup` в высоконагруженных сценариях. Функция выполняет сетевые запросы, используя однопоточный процесс, который блокирует поток выполнения при каждом запросе, пока не получит ответ, из-за чего производительность в высоконагруженных приложениях может деградировать.

Избегайте использования этой функции в контекстах, чувствительных к задержке, из-за возможных сетевых задержек или сбоев.

Спецификация функции

```
dns_lookup(value: <строка>, [qtype: <строка>, qclass: <строка>, options: <объект>])
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Доменное имя.		да
`qtype`	строка	Тип искомой записи DNS.		нет
`qclass`	строка	Класс сети, для которой выполняется поиск DNS-записей.	`IN`	нет
`options`	строка	Опции поиска.		нет

Ошибки

Функция `dns_lookup` **может возвращать ошибки**, для которых требуется обработка:

- Невозможно разобрать имя хоста `value`.
- Невозможно разобрать тип запроса `qtype`.
- Невозможно разобрать класс сети `qclass`.
- Не удалось выполнить запрос DNS Lookup.

Примеры

Базовый поиск

Пример 392. Исходный код

```
res = dns_lookup!("dns.google")
# сбросить нестатический ttl, так что результат статический
res.answers = map_values(res.answers) -> |value| {
value.ttl = 600
value
}
# удалить дополнительные ответы
```

```
res.answers = filter(res.answers) -> |_, value| {
value.rData == "8.8.8.8"
}
# удалить класс, поскольку он динамический
res.additional = map_values(res.additional) -> |value| {
del(value.class)
value
}
res
```

Пример 393. Результат

```
{
  "additional": [
    {
      "domainName": "",
      "rData": "OPT ...",
      "recordType": "OPT",
      "recordTypeId": 41,
      "ttl": 0
    }
  ],
  "answers": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "rData": "8.8.8.8",
      "recordType": "A",
      "recordTypeId": 1,
      "ttl": 600
    }
  ],
  "authority": [],
  "fullRcode": 0,
  "header": {
    "aa": false,
    "ad": false,
    "anCount": 2,
    "arCount": 1,
    "cd": false,
    "nsCount": 0,
    "opcode": 0,
    "qdCount": 1,
    "qr": true,
    "ra": true,
    "rcode": 0,
    "rd": true,
    "tc": false
  },
  "question": [
```

```
    {
      "class": "IN",
      "domainName": "dns.google",
      "questionType": "A",
      "questionTypeId": 1
    }
  ],
  "rcodeName": "NOERROR"
}
```

Заданные `qclass` и `qtype`

Пример 394. Исходный код

```
res = dns_lookup!("dns.google", class: "IN", qtype: "A")
# сбросить нестатический ttl, так что результат статический
res.answers = map_values(res.answers) -> |value| {
  value.ttl = 600
  value
}
#удалить дополнительные ответы
res.answers = filter(res.answers) -> |_, value| {
    value.rData == "8.8.8.8"
}
# удалить класс, поскольку он динамический
res.additional = map_values(res.additional) -> |value| {
    del(value.class)
    value
}
res
```

Пример 395. Результат

```
{
  "additional": [
    {
      "domainName": "",
      "rData": "OPT ...",
      "recordType": "OPT",
      "recordTypeId": 41,
      "ttl": 0
    }
  ],
  "answers": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "rData": "8.8.8.8",
      "recordType": "A",
      "recordTypeId": 1,
```

```
      "ttl": 600
    }
  ],
  "authority": [],
  "fullRcode": 0,
  "header": {
    "aa": false,
    "ad": false,
    "anCount": 2,
    "arCount": 1,
    "cd": false,
    "nsCount": 0,
    "opcode": 0,
    "qdCount": 1,
    "qr": true,
    "ra": true,
    "rcode": 0,
    "rd": true,
    "tc": false
  },
  "question": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "questionType": "A",
      "questionTypeId": 1
    }
  ],
  "rcodeName": "NOERROR"
}
```

Заданные опции

Пример 396. Исходный код

```
res = dns_lookup!("dns.google", options: {"timeout": 30, "attempts": 5})
# сбросить нестатический ttl, так что результат статический
res.answers = map_values(res.answers) -> |value| {
value.ttl = 600
value
}
# удалить дополнительные ответы
res.answers = filter(res.answers) -> |_, value| {
value.rData == "8.8.8.8"
}
# удалить класс, поскольку он динамический
res.additional = map_values(res.additional) -> |value| {
del(value.class)
value
}
```

```
res
```

Пример 397. Результат


```
{
  "additional": [
    {
      "domainName": "",
      "rData": "OPT ...",
      "recordType": "OPT",
      "recordTypeId": 41,
      "ttl": 0
    }
  ],
  "answers": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "rData": "8.8.8.8",
      "recordType": "A",
      "recordTypeId": 1,
      "ttl": 600
    }
  ],
  "authority": [],
  "fullRcode": 0,
  "header": {
    "aa": false,
    "ad": false,
    "anCount": 2,
    "arCount": 1,
    "cd": false,
    "nsCount": 0,
    "opcode": 0,
    "qdCount": 1,
    "qr": true,
    "ra": true,
    "rcode": 0,
    "rd": true,
    "tc": false
  },
  "question": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "questionType": "A",
      "questionTypeId": 1
    }
  ],
  "rcodeName": "NOERROR"
```

```
}
```

Заданный DNS-сервер

Пример 398. Исходный код

```
res = dns_lookup!("dns.google", options: {"servers": ["dns.quad9.net"]})
# сбросить нестатический ttl, так что результат статический
res.answers = map_values(res.answers) -> |value| {
value.ttl = 600
value
}
# удалить дополнительные ответы
res.answers = filter(res.answers) -> |_, value| {
value.rData == "8.8.8.8"
}
# удалить класс, поскольку он динамический
res.additional = map_values(res.additional) -> |value| {
del(value.class)
value
}
res
```

Пример 399. Результат

```
{
  "additional": [
  {
    "domainName": "",
    "rData": "OPT ...",
    "recordType": "OPT",
    "recordTypeId": 41,
    "ttl": 0
  }
  ],
  "answers": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "rData": "8.8.8.8",
      "recordType": "A",
      "recordTypeId": 1,
      "ttl": 600
    }
  ],
  "authority": [],
  "fullRcode": 0,
  "header": {
  "aa": false,
  "ad": false,
```

```
  "anCount": 2,
  "arCount": 1,
  "cd": false,
  "nsCount": 0,
  "opcode": 0,
  "qdCount": 1,
  "qr": true,
  "ra": true,
  "rcode": 0,
  "rd": true,
  "tc": false
  },
  "question": [
    {
      "class": "IN",
      "domainName": "dns.google",
      "questionType": "A",
      "questionTypeId": 1
    }
  ],
  "rcodeName": "NOERROR"
}
```

parse_glog

Разбирает `value` с использованием формата glog (Google Logging Library).

Спецификация функции

```
parse_glog(value: <строка>)
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да

Ошибки

Функция `parse_glog` **может возвращать ошибки**, для которых требуется обработка:

- `value` не соответствует формату `glog`.

Примеры

Разбор через формат `glog`

Пример 400. Исходный код

```
parse_glog!("I20210131 14:48:54.411655 15520 main.c++:9] Hello world!")
```

Пример 401. Результат

```
{
  "file": "main.c++",
  "id": 15520,
  "level": "info",
  "line": 9,
  "message": "Hello world!",
  "timestamp": "2021-01-31T14:48:54.411655Z"
}
```

parse_grok

Разбирает `value`, используя формат `grok`. Поддерживаются все шаблоны grok.

Рекомендуется использовать шаблоны Grok, которые поддерживаются сообществом. Такие шаблоны, скорее всего, будут проверены и улучшены в будущем, в отличие от пользовательских шаблонов.

Спецификация функции

```
parse_grok(value: <строка>, pattern: <строка>)
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`pattern`	строка	Шаблон Grok.		да

Ошибки

Функция `parse_grok` **может возвращать ошибки**, для которых требуется обработка:

- `value` не удается разобрать с использованием предоставленного шаблона `pattern`.

Примеры

Разбор с использованием Grok

Пример 402. Исходный код

```
parse_grok!(
"2020-10-02T23:22:12.223222Z info Hello world",
"%{TIMESTAMP_ISO8601:timestamp} %{LOGLEVEL:level} %{GREEDYDATA:message}"
)
```

Пример 403. Результат

```
{
  "level": "info",
  "message": "Hello world",
  "timestamp": "2020-10-02T23:22:12.223222Z"
}
```

parse_groks

Разбирает `value`, используя несколько шаблонов `grok`. Поддерживаются все шаблоны grok.

Рекомендуется использовать шаблоны Grok, которые поддерживаются сообществом. Такие шаблоны, скорее всего, будут проверены и улучшены в будущем, в отличие от пользовательских шаблонов.

Спецификация функции

```
parse_groks(value: <строка>, patterns: <массив>, [aliases: <объект>, alias_sources:
<массив>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`patterns`	массив	Шаблоны Grok, которые применяются последовательно до первого совпадения.		да
`aliases`	объект	Общий набор псевдонимов Grok, на которые можно ссылаться в шаблонах для их упрощения.		нет
`alias_sources`	массив строк	Пути к файлам, содержащим псевдонимы в формате JSON.		нет

Ошибки

Функция `parse_groks` **может возвращать ошибки**, для которых требуется обработка:

- `value` не удается разобрать с использованием предоставленных шаблонов;
- `patterns` не является массивом;
- `aliases` не является объектом;
- элемент массива `alias_sources` не является строкой или не указывает на корректный файл.

Примеры

Разбор с использованием нескольких шаблонов Grok

Пример 404. Исходный код

```
parse_groks!(
"2020-10-02T23:22:12.223222Z info Hello world",
patterns: [
"%{common_prefix} %{_status} %{_message}",
"%{common_prefix} %{_message}",
],
aliases: {
"common_prefix": "%{_timestamp} %{_loglevel}",
"_timestamp": "%{TIMESTAMP_ISO8601:timestamp}",
"_loglevel": "%{LOGLEVEL:level}",
"_status": "%{POSINT:status}",
"_message": "%{GREEDYDATA:message}"
})
```

Пример 405. Результат

```
{
  "level": "info",
  "message": "Hello world",
  "timestamp": "2020-10-02T23:22:12.223222Z"
}
```

parse_influxdb

Разбирает `value` как строку протокола строк InfluxDB, создавая список метрик, совместимых с Vector.

Спецификация функции

```
parse_influxdb(value: <строка> )
:: <массив> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление протокола строк InfluxDB для разбора.		да

Примечания

У этой функции есть особенности поведения.

- Эта функция вернет событие журнала с размерностью Vector-совместимой метрики, но не само событие метрики. Вероятно, вам потребуется обработать вывод этой функции преобразованием `log_to_metric` с опцией `all_metrics`, установленной в значение `true`, чтобы конвертировать события логов с размерностью метрик в события метрик и создать реальные метрики.
- Единственный тип метрики, который создается — `gauge`. К имени каждой метрики добавляется префикс, состоящий из поля `measurement`, подчеркивания (_) и поля `field key`.

- строка — это единственный тип, который не поддерживается как значение поля, из-за ограничений модели метрик Vector.

Ошибки

Функция `parse_influxdb` **может возвращать ошибки**, для которых требуется обработка:

- `value` — не валидная строка протокола строк InfluxDB.
- Список полей содержит поле типа `строка`.
- Список полей содержит значение поля `NaN`.

Примеры

Разбор протокола строк InfluxDB

Пример 406. Исходный код

```
parse_influxdb!("cpu,host=A,region=us-west
usage_system=64i,usage_user=10u,temperature=50.5,on=true,sleep=false
1590488773254420000")
```

Пример 407. Результат

```
[
  {
    "gauge": {
      "value": 64
    },
    "kind": "absolute",
    "name": "cpu_usage_system",
    "tags": {
      "host": "A",
      "region": "us-west"
    },
    "timestamp": "2020-05-26T10:26:13.254420Z"
  },
  {
    "gauge": {
      "value": 10
    },
    "kind": "absolute",
    "name": "cpu_usage_user",
    "tags": {
      "host": "A",
      "region": "us-west"
    },
    "timestamp": "2020-05-26T10:26:13.254420Z"
  },
  {
    "gauge": {
```

```
      "value": 50.5
    },
    "kind": "absolute",
    "name": "cpu_temperature",
    "tags": {
      "host": "A",
      "region": "us-west"
    },
    "timestamp": "2020-05-26T10:26:13.254420Z"
  },
  {
    "gauge": {
      "value": 1
    },
    "kind": "absolute",
    "name": "cpu_on",
    "tags": {
      "host": "A",
      "region": "us-west"
    },
    "timestamp": "2020-05-26T10:26:13.254420Z"
  },
  {
    "gauge": {
      "value": 0
    },
    "kind": "absolute",
    "name": "cpu_sleep",
    "tags": {
      "host": "A",
      "region": "us-west"
    },
    "timestamp": "2020-05-26T10:26:13.254420Z"
  }
]
```

parse_klog

Разбирает `value` в формате klog, используемом компонентами Kubernetes.

Спецификация функции

```
parse_klog(value: <строка>)
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да

Ошибки

Функция `parse_klog` **может возвращать ошибки**, для которых требуется обработка:

- `value` не соответствует формату `klog`.

Примеры

Разбор по формату klog

Пример 408. Исходный код

```
parse_klog!("I0505 17:59:40.692994 28133 klog.go:70] hello from klog")
```

Пример 409. Результат

```
{
  "file": "klog.go",
  "id": 28133,
  "level": "info",
  "line": 70,
  "message": "hello from klog",
  "timestamp": "2024-05-05T17:59:40.692994Z"
}
```

rv_parse_leef

Разбирает `value` в формате Log Event Extended Format (LEEF).

Спецификация функции

```
rv_parse_leef(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление сообщения LEEF		да

Ошибки

Функция `rv_parse_leef` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректным сообщением LEEF.

Пример

Сообщение LEEF с заголовком syslog и атрибутами события, разделенными символом ^

Пример 410. Исходный код

```
rv_parse_leef!("<13>1 2024-07-18T11:07:53.520Z 192.168.1.1
LEEF:2.0|Microsoft|MSExchange|2007|7732|^|dvchost=www.example.com^msg=io\t|p")
```

Пример 411. Результат

```
{
  "syslog_facility": "1",
  "syslog_priority": "13",
  "syslog_severity": "5",
  "at": "2024-07-18T11:07:53.520Z",
  "ahost": "192.168.1.1",
  "dvchost": "www.example.com",
  "msg": "io\t|p",
  "deviceVendor": "Microsoft",
  "eventName": "7732",
  "productName": "MSExchange",
  "productVersion": "2007",
}
```

parse_linux_authorization

Разбирает журналы авторизации Linux, которые обычно находятся в каталогах `/var/log/auth.log` (в системах на основе Debian) или `/var/log/secure` (в системах на основе RedHat), в соответствии с форматом syslog.

Спецификация функции

```
parse_linux_authorization(value: <строка>)
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Текст, содержащий сообщение для разбора.		да

Примечания

Эта функция имеет особое поведение: если в сообщении не указан год, он определяется автоматически:

- Если текущий месяц — январь, а сообщение относится к декабрю, то используется предыдущий год.
- В противном случае используется текущий год.

Ошибки

Функция `parse_linux_authorization` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является правильно отформатированным сообщением syslog.

Примеры

Разбор события авторизации Linux

Пример 412. Исходный код

```
parse_linux_authorization!(
s'Mar 23 01:49:58 localhost sshd[1111]: Accepted publickey for eng from
198.51.100.1 port 8888 ssh2: RSA SHA256:foobar'
)
```

Пример 413. Результат

```
{
  "appname": "sshd",
  "hostname": "localhost",
  "message": "Accepted publickey for eng from 198.51.100.1 port 8888 ssh2: RSA
SHA256:foobar",
  "procid": 1111,
  "timestamp": "2023-03-23T01:49:58Z"
}
```

parse_logfmt

Разбирает `value` в формате logfmt.

- Ключи и значения могут быть заключены в двойные кавычки `"`.
- Символ `"` может быть экранирован символом `\`.
- Согласно спецификации logfmt, ключами без значений присваивается логическое значение `true`.
- Функция обрабатывает три управляющие последовательности при разборе: `\n`, `\"` и `\\`. Это значит, что, например, `\n` во входной строке будет заменен на действительный символ новой строки в разобранных ключах или значениях.

Спецификация функции

```
parse_logfmt(value: <строка>)
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да

Ошибки

Функция `parse_logfmt` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является правильно отформатированной строкой "ключ — значение" (logfmt).

Примеры

Разбор лога в формате logfmt

Пример 414. Исходный код

```
parse_logfmt!(
"@timestamp=\"Sun Jan 10 16:47:39 EST 2021\" level=info msg=\"Stopping all fetchers\"
tag#production=stopping_fetchers id=ConsumerFetcherManager-1382721708341
module=kafka.consumer.ConsumerFetcherManager"
)
```

Пример 415. Результат

```
{
  "@timestamp": "Sun Jan 10 16:47:39 EST 2021",
  "id": "ConsumerFetcherManager-1382721708341",
  "level": "info",
  "module": "kafka.consumer.ConsumerFetcherManager",
  "msg": "Stopping all fetchers",
  "tag#production": "stopping_fetchers"
}
```

rv_parse_msgpack

Разбирает `value` в формате MessagePack.

Спецификация функции

```
rv_parse_msgpack(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление MessagePack		да

Примечания

Особенности функции, о которых нужно знать:

- Возвращаются только типы MessagePack. Дополнительные типы (extension) не поддерживаются. Если нужно преобразовать строку в метку времени, используйте функцию `parse_timestamp`.
- В типе map поддерживаются только строковые ключи.

Ошибки

Функция `rv_parse_msgpack` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является массивом байтов.

Примеры

Преобразование массива байтов

Пример 416. Исходный код

```
# .message -- массив байтов [135, 163, 105, 110, 116, 1, 165, 102, 108, 111, 97, 116,
203, 63, 224, 0, 0, 0, 0, 0, 0, 167, 98, 111, 111, 108, 101, 97, 110, 195, 164, 110,
117, 108, 108, 192, 166, 115, 116, 114, 105, 110, 103, 167, 102, 111, 111, 32, 98, 97,
114, 165, 97, 114, 114, 97, 121, 146, 163, 102, 111, 111, 163, 98, 97, 114, 166, 111,
98, 106, 101, 99, 116, 130, 163, 102, 111, 111, 1, 163, 98, 97, 122, 203, 63, 224, 0,
0, 0, 0, 0, 0,]
rv_parse_msgpack!(.message)
```

Пример 417. Результат

```
{
  "int": 1,
  "float": 0.5,
  "boolean": true,
  "null": null,
  "string": "foo bar",
  "array": [
    "foo",
    "bar"
  ],
  "object": {
    "foo": 1,
    "baz": 0.5
}
```

Преобразование после декодирования из Base64

Пример 418. Исходный код

```
rv_parse_msgpack!(decode_base64!("3wAAAAGlZmllbGSldmFsdWU="))
```

Пример 419. Результат

```
{
  "field": "value"
}
```

rv_parse_msgpack_batch

Разбирает `value`, которое состоит из нескольких сообщений в формате MessagePack.

Спецификация функции

```
rv_parse_msgpack_batch(value: <строка>)
:: <массив объектов>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление пакета объектов MessagePack		да

Примечания

Особенности функции, о которых нужно знать:

- Возвращаются только типы MessagePack. Дополнительные типы (extension) не поддерживаются. Если нужно преобразовать строку в метку времени, используйте функцию `parse_timestamp`.
- В типе map поддерживаются только строковые ключи.

Ошибки

Функция `rv_parse_msgpack_batch` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является массивом байтов.

Примеры

Преобразование массива байтов

Пример 420. Исходный код

```
# .message -- массив байтов [135, 163, 105, 110, 116, 1, 165, 102, 108, 111, 97, 116,
203, 63, 224, 0, 0, 0, 0, 0, 0, 167, 98, 111, 111, 108, 101, 97, 110, 195, 164, 110,
117, 108, 108, 192, 166, 115, 116, 114, 105, 110, 103, 167, 102, 111, 111, 32, 98, 97,
114, 165, 97, 114, 114, 97, 121, 146, 163, 102, 111, 111, 163, 98, 97, 114, 166, 111,
98, 106, 101, 99, 116, 130, 163, 102, 111, 111, 1, 163, 98, 97, 122, 203, 63, 224, 0,
0, 0, 0, 0, 0, 135, 163, 105, 110, 116, 1, 165, 102, 108, 111, 97, 116, 203, 63, 224,
0, 0, 0, 0, 0, 0, 167, 98, 111, 111, 108, 101, 97, 110, 195, 164, 110, 117, 108, 108,
192, 166, 115, 116, 114, 105, 110, 103, 167, 102, 111, 111, 32, 98, 97, 114, 165, 97,
114, 114, 97, 121, 146, 163, 102, 111, 111, 163, 98, 97, 114, 166, 111, 98, 106, 101,
99, 116, 130, 163, 102, 111, 111, 1, 163, 98, 97, 122, 203, 63, 224, 0, 0, 0, 0, 0,
0,]
rv_parse_msgpack_batch!(.message)
```

Пример 421. Результат

```
[
```

```
  {
    "int": 1,
    "float": 0.5,
    "boolean": true,
    "null": null,
    "string": "foo bar",
    "array": [
      "foo",
      "bar"
    ],
    "object": {
      "foo": 1,
      "baz": 0.5
    }
  },
  {
    "int": 1,
    "float": 0.5,
    "boolean": true,
    "null": null,
    "string": "foo bar",
    "array": [
      "foo",
      "bar"
    ],
    "object": {
      "foo": 1,
      "baz": 0.5
    }
  }
]
```

Преобразование после декодирования из Base64

Пример 422. Исходный код

```
rv_parse_msgpack_batch(decode_base64!("3wAAAAGlZmllbGSldmFsdWXfAAAAAaVmaWVsZKV2YWx1ZQ==
="))
```

Пример 423. Результат

```
[
  {
    "field": "value"
  },
  {
    "field": "value"
  }
]
```

rv_parse_netflow

Разбирает `value` в формате пакетов NetFlow.

Спецификация функции

```
rv_parse_netflow(value: <строка>)
:: <массив объектов>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление пакетов NetFlow.		да

Ошибки

Функция `rv_parse_netflow` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является правильным NetFlow пакетом

Примеры

Разбор пакетов NetFlow

Пример 424. Исходный код

```
rv_parse_netflow!(decode_base64!("AAUCAAMABAAFAAYHCAkAAQIDBAUGBwgJAAECAwQFBgcICQABAgME
BQYHCAkAAQIDBAUGBwgJAAECAwQFBgcICQABAgMEBQYH"))
```

Пример 425. Результат

```
[
  {
    "V5": {
      "body": {
        "d_octets": 66051,
        "d_pkts": 101124105,
        "dst_addr": "4.5.6.7",
        "dst_as": 515,
        "dst_mask": 5,
        "dst_port": 1029,
        "first": {
          "nanos": 87000000,
          "secs": 67438
        },
        "input": 515,
        "last": {
          "nanos": 553000000,
          "secs": 134807
        },
```

```
        "next_hop": "8.9.0.1",
        "output": 1029,
        "pad1": 6,
        "pad2": 1543,
        "protocol_number": 8,
        "protocol_type": "EGP",
        "src_addr": "0.1.2.3",
        "src_as": 1,
        "src_mask": 4,
        "src_port": 515,
        "tcp_flags": 7,
        "tos": 9
      },
      "header": {
        "count": 512,
        "engine_id": 7,
        "engine_type": 6,
        "flow_sequence": 33752069,
        "sampling_interval": 2057,
        "sys_up_time": {
          "nanos": 672000000,
          "secs": 50332
        },
        "unix_nsecs": 134807553,
        "unix_secs": 83887623,
        "version": 5
      }
    }
  }
]
```


parse_nginx_log

Разбирает строки журналов доступа и ошибок Nginx. Строки могут быть в формате `combined`, `ingress_upstreaminfo`, `main` или `error`.

Спецификация функции

```
parse_nginx_log(value: <строка>, format: <строка>, [timestamp_format: <строка>])
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`format`	строка	Формат, используемый для разбора журнала.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`timestamp_format`	строка	Формат даты/времени, используемый для кодирования метки времени. Если в метке времени не указан часовой пояс, время разбирается как локальное. Формат по умолчанию `%d/%b/%Y:%T %z` для комбинированных журналов и `%Y/%m/%d %H:%M:%S` для журналов ошибок.	`%d/%b/%Y:%T %z`	нет

Примечания

Эта функция имеет особое поведение, о котором следует знать.

- Отсутствующая информация в сообщении может быть обозначена `-`. Эти поля опускаются в результате.

Ошибки

Функция `parse_nginx_log` **может возвращать ошибки**, для которых требуется обработка:

- `value` не соответствует указанному формату.
- `timestamp_format` не является допустимой строкой формата.
- Метка времени в `value` не удается разобрать с использованием предоставленного `timestamp_format`.

Примеры

Разбор в формате журнала Nginx (комбинированном)

Пример 426. Исходный код

```
parse_nginx_log!(
s'203.0.113.1 - alice [01/Apr/2021:12:02:31 +0000] "POST /not-found HTTP/1.1" 404 153
"http://localhost/somewhere" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/72.0.3626.119 Safari/537.36" "2.75"',
"combined",
)
```

Пример 427. Результат

```
{
  "client": "203.0.113.1",
  "user": "alice",
  "timestamp": "2021-04-01T12:02:31Z",
  "request": "POST /not-found HTTP/1.1",
  "method": "POST",
  "path": "/not-found",
  "protocol": "HTTP/1.1",
  "status": 404,
```

```
  "size": 153,
  "referer": "http://localhost/somewhere",
  "agent": "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/72.0.3626.119 Safari/537.36",
  "compression": "2.75"
}
```

Разбор в формате журнала ошибок Nginx

Пример 428. Исходный код

```
parse_nginx_log!(
s'2021/04/01 13:02:31 [error] 31#31: *1 open() "/usr/share/nginx/html/not-found"
failed (2: No such file or directory), client: 203.0.113.1, server: localhost,
request: "POST /not-found HTTP/1.1", host: "localhost:8081"',
"error"
)
```

Пример 429. Результат

```
{
  "timestamp": "2021-04-01T13:02:31Z",
  "severity": "error",
  "pid": 31,
  "tid": 31,
  "cid": 1,
  "message": "open() \"/usr/share/nginx/html/not-found\" failed (2: No such file or
directory)",
  "client": "203.0.113.1",
  "server": "localhost",
  "request": "POST /not-found HTTP/1.1",
  "host": "localhost:8081"
}
```

parse_proto

Разбирает `value` как payload — полезную нагрузку Protocol Buffers.

Спецификация функции

```
parse_proto(value: <строка>, desc_file: <строка>, message_type: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Полезная нагрузка Protocol Buffers для разбора.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`desc_file`	строка	Путь к файлу набора дескрипторов Protocol Buffers. Должен быть строковым литералом. Этот файл является результатом работы команды `protoc -o`.		да
`message_type`	строка	Имя типа сообщения, используемого для сериализации. Должно быть строковым литералом.		да

Примечание

Функция разбирает и возвращает только сообщения (message) Protocol Buffers.

Ошибки

Функция `encode_proto` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является валидным payload Protocol Buffers.
- Файл, указанный в аргументе `desc_file`, не существует.
- Тип сообщения, указанный в аргументе `message_type`, отсутствует в файле дескрипторов.

Пример

Пример 430. Исходный код

```
.payload = encode_base64(encode_proto!({"name": "someone", "phones": [{"number":
"123456"}]}, "resources/protobuf_descriptor_set.desc", "test_protobuf.Person"))
```

Пример 431. Результат

```
Cgdzb21lb25lIggKBjEyMzQ1Ng==
```

rv_parse_qradar

Разбирает `value` в формате QRadar, включая необязательный заголовок syslog.

Спецификация функции

```
rv_parse_qradar(value: <строка>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление сообщения QRadar		да

Ошибки

Функция `rv_parse_qradar` **может возвращать ошибки**, для которых требуется обработка:

- `value` закодировано не в UTF-8;
- `value` содержит ключ без соответствующего значения.

Примеры

Разбор сообщения без заголовка syslog и со вложенным полем Message

Пример 432. Исходный код

```
rv_parse_msgpack!("AgentDevice=WindowsLog\tAgentLogFile=Security\tPluginVersion=7.2.9.
105\tSource=Microsoft-Windows-Security-
Auditing\tComputer=dc04.lab2012.local\tOriginatingComputer=192.0.2.0\tUser=\tDomain=\t
EventID=4624\tEventIDCode=4624\tEventType=8\tEventCategory=12544\tRecordNumber=1110397
1\tTimeGenerated=1610705793\tTimeWritten=1610705793\tLevel=Log Always\tKeywords=Audit
Success\tTask=SE_ADT_LOGON_LOGON\tOpcode=Info\tMessage=Вход с учетной записью выполнен
успешно.  Субъект:  ИД безопасности:  NULL SID  Имя учетной записи:  -  Домен учетной
записи:  -  Код входа:  0x0  Тип входа:  3  Уровень олицетворения:  Делегирование
Новый вход:  ИД безопасности:  NT AUTHORITY\\СИСТЕМА  Имя учетной записи:  USER  Домен
учетной записи:  LAB2012  Код входа:  0x1E6112B9  GUID входа:  {06457096-03A4-55FB-
28EA-9882B565E681}  Сведения о процессе:  Идентификатор процесса:  0x0  Имя процесса:
-  Сведения о сети:  Имя рабочей станции: -  Сетевой адрес источника:
fe80::7830:1b00:4e5f:1300  Порт источника:  51755  Данное событие возникает при
создании сеанса входа.\n".as_bytes())
```

Пример 433. Результат

```
{
  "AgentDevice": "WindowsLog",
  "AgentLogFile": "Security",
  "PluginVersion": "7.2.9.105",
  "Source": "Microsoft-Windows-Security-Auditing",
  "Computer": "dc04.lab2012.local",
  "OriginatingComputer": "192.0.2.0"
  "User": ""
  "Domain": ""
  "EventID": "4624"
  "EventIDCode": "4624"
  "EventType": "8"
  "EventCategory": "12544"
  "RecordNumber": "11103971"
  "TimeGenerated": "1610705793"
  "TimeWritten": "1610705793"
  "Level": "Log Always"
  "Keywords": "Audit Success"
  "Task": "SE_ADT_LOGON_LOGON"
  "Opcode": "Info"
  "Message": {
```

```
    "Вход с учетной записью выполнен успешно.": "",
    "Субъект": "ИД безопасности: NULL SID",
    "Имя учетной записи": "-",
    "Домен учетной записи": "-",
    "Код входа": "0x0",
    "Тип входа": "3",
    "Уровень олицетворения": "Делегирование",
    "Новый вход": {
      "ИД безопасности": "NT AUTHORITY\\СИСТЕМА",
      "Имя учетной записи": "USER",
      "Домен учетной записи": "LAB2012",
      "Код входа": "0x1E6112B9",
      "GUID входа": "{06457096-03A4-55FB-28EA-9882B565E681}"
    },
    "Сведения о процессе": {
      "Идентификатор процесса": "0x0",
      "Имя процесса": "-"
    },
    "Сведения о сети": {
      "Имя рабочей станции": "-",
      "Сетевой адрес источника": "2001:DB8::7830:1b00:4e5f:1300",
      "Порт источника": "51755"
    }
  }
}
```


Разбор сообщения с заголовком syslog

Пример 434. Исходный код

```
rv_parse_msgpack!("<13>Jan 15 13:16:35 DC04
AgentDevice=WindowsLog\tAgentLogFile=Security\tPluginVersion=7.2.9.105\tSource=Microso
ft-Windows-Security-
Auditing\tComputer=dc04.lab2012.local\tOriginatingComputer=192.0.2.0\tUser=\tDomain=\t
EventID=4624\tEventIDCode=4624\tEventType=8\tEventCategory=12544\tRecordNumber=1110397
1\tTimeGenerated=1610705793\tTimeWritten=1610705793\tLevel=Log Always\tKeywords=Audit
Success\tTask=SE_ADT_LOGON_LOGON\tOpcode=Info\n".as_bytes())
```

Пример 435. Результат


```
{
  "syslog_facility": "1",
  "syslog_priority": "13",
  "syslog_severity": "5",
  "at": "Jan 15 13:16:35",
  "ahost": "DC04",
  "AgentDevice": "WindowsLog",
  "AgentLogFile": "Security",
  "PluginVersion": "7.2.9.105",
  "Source": "Microsoft-Windows-Security-Auditing",
```

```
  "Computer": "dc04.lab2012.local",
  "OriginatingComputer": "192.0.2.0"
  "User": ""
  "Domain": ""
  "EventID": "4624"
  "EventIDCode": "4624"
  "EventType": "8"
  "EventCategory": "12544"
  "RecordNumber": "11103971"
  "TimeGenerated": "1610705793"
  "TimeWritten": "1610705793"
  "Level": "Log Always"
  "Keywords": "Audit Success"
  "Task": "SE_ADT_LOGON_LOGON"
  "Opcode": "Info"
}
```

parse_syslog

Разбирает `value` в формате syslog.

Спецификация функции

```
parse_syslog(value: <строка>)
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Текст, содержащий сообщение syslog для разбора.		да

Примечания

Эта функция имеет особое поведение, о котором следует знать.

- Функция разбирает различные форматы syslog, такие как RFC 6587, RFC 5424, RFC 3164, стиль syslog в Nginx и другие распространенные варианты.
- Все значения возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.

Ошибки

Функция `parse_syslog` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является правильно отформатированным сообщением syslog.

Примеры

Разбор журнала Syslog (5424)

Пример 436. Исходный код

```
parse_syslog!(
s'<13>1 2020-03-13T20:45:38.119Z example.name non 2426 ID931 [exampleSDID@32473
iut="3" eventSource= "Application" eventID="1011"] Try to override the THX port, maybe
it will reboot the neural interface!'
)
```

Пример 437. Результат

```
{
  "severity": "notice",
  "facility": "user",
  "timestamp": "2020-03-13T20:45:38.119Z",
  "hostname": "example.name",
  "appname": "non",
  "procid": 2426,
  "msgid": "ID931",
  "message": "Try to override the THX port, maybe it will reboot the neural
interface!",
  "exampleSDID@32473": {
    "eventID": "1011",
    "eventSource": "Application",
    "iut": "3"
  },
  "version": 1
}
```

Функции преобразования кодов и параметров Syslog

to_syslog_facility

Преобразует `value` из кода устройства Syslog в соответствующее ему ключевое слово Syslog, например, `0` в `"kern"`, а `1` — в `"user"`.

Спецификация функции

```
to_syslog_facility(value: <целое число>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число	Код устройства.		да

Ошибки

Функция `to_syslog_facility` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является допустимым кодом устройства Syslog.

Примеры

Приведение к устройству Syslog

Пример 438. Исходный код

```
to_syslog_facility!(4)
```

Пример 439. Результат

```
auth
```

to_syslog_facility_code

Преобразует `value` из ключевого слова Syslog в соответствующий ему код устройства Syslog (от `0` до `23`).

Спецификация функции

```
to_syslog_facility_code(value: <строка>)
:: <целое число>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Ключевое слово Syslog.		да

Ошибки

Функция `to_syslog_facility_code` **возвращает ошибку**, если `value` не является допустимым ключевым словом Syslog.

Пример

Приведение к ключевому слову Syslog

Пример 440. Исходный код

```
to_syslog_facility_code!("authpriv")
```

Пример 441. Результат

```
10
```

to_syslog_level

Преобразует `value` из кода уровня серьезности Syslog в соответствующее ключевое слово,

например, `0` в `"emerg"`, `1` — в `"alert"`.

Спецификация функции

```
to_syslog_level(value: <целое число>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число	Код уровня серьезности.		да

Ошибки

Функция `to_syslog_level` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является допустимым кодом уровня серьезности Syslog.

Примеры

Приведение к уровню Syslog

Пример 442. Исходный код

```
to_syslog_level!(5)
```

Пример 443. Результат

```
notice
```

to_syslog_severity

Преобразует `value`, ключевое слово уровня Syslog, в целое число уровня серьезности Syslog (`0` до `7`).

Спецификация функции

```
to_syslog_severity(value: <строка>)
:: <целое число> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Ключевое слово уровня Syslog для преобразования.		да

Ошибки

Функция `to_syslog_severity` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является допустимым ключевым словом уровня Syslog.

Примеры

Приведение к уровню серьезности Syslog

Пример 444. Исходный код

```
to_syslog_severity!("alert")
```

Пример 445. Результат

```
1
```

Функции разбора структурированных данных

parse_cbor

Разбирает `value` в формате CBOR.

Спецификация функции

```
parse_cbor(value: <строка>)
:: <двоичное значение | целое число | число с плавающей точкой | строка | объект |
массив | пустое значение>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Полезная нагрузка CBOR для разбора.		да

Функция возвращает только типы CBOR.

Ошибки

Функция `parse_csv` **возвращает ошибки**, для которых требуется обработка, если `value` не является допустимой полезной нагрузкой CBOR.

Пример

Разбор CBOR

Пример 446. Исходный код

```
parse_cbor!(decode_base64!("oWVmaWVsZGV2YWx1ZQ=="))
```

Пример 447. Результат

```
{
```

```
  "field": "value"
}
```

parse_csv

Разбирает отдельную строку в формате CSV. Если входное значение состоит из нескольких строк, будет проанализирована только первая строка.

Спецификация функции

```
parse_csv(value: <строка>, [delimiter: <строка>])
:: <массив> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`delimiter`	строка	Разделитель полей для использования при разборе. Должен быть однобайтовым символом UTF-8.	`,`	нет

Примечания

У этой функции есть специальное поведение, которое следует учитывать:

- Все значения возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.

Ошибки

Функция `parse_csv` **может возвращать ошибки**, для которых требуется обработка:

- Разделитель не является однобайтовым символом UTF-8.
- `value` не является допустимой строкой CSV.

Примеры

Разбор отдельной строки в формате CSV

Пример 448. Исходный код

```
parse_csv!("foo,bar,\"foo \"\", bar\"")
```

Пример 449. Результат

```
[
  "foo",
  "bar",
  "foo \", bar"
```

```
]
```

Разбор отдельной строки в формате CSV с пользовательским разделителем

Пример 450. Исходный код

```
parse_csv!("foo bar", delimiter: " ")
```

Пример 451. Результат

```
[
  "foo",
  "bar"
]
```

rv_parse_html

Разбирает `value` в формате HTML.

Спецификация функции

```
rv_parse_html(value: <строка>, [trim: <логическое значение>, attr_prefix: <строка>,
text_key: <строка>, always_use_text_key: <логическое значение>, parse_bool:
<логическое значение>, parse_null: <логическое значение>, parse_number: <логическое
значение>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	HTML-документ в виде строки.		да
`trim`	логическое значение	Удалять пробелы вокруг элементов HTML.	`true`	нет
`include_attr`	логическое значение	Включать атрибуты HTML в возвращаемый объект.	`true`	нет
`attr_prefix`	строка	Префикс строки для ключей атрибутов HTML.	`"@"`	нет
`text_key`	строка	Ключ, под которым возвращается содержимое элементов HTML.	`"text"`	нет
`always_use_text_key`	логическое значение	Всегда возвращать содержимое элементов HTML под ключом, указанным в параметре `text_key`. Если `true`, содержимое будет сохраняться под отдельным ключом, даже если у элемента нет атрибутов.	`false`	нет

Аргумент	Тип	Описание	По умолчанию	Обязателен
`parse_bool`	логическое значение	Интерпретировать строки `"true"` и `"false"` как логические значения.	`true`	нет
`parse_null`	логическое значение	Интерпретировать пустые строки и строки `"null"` как пустые значения.	`true`	нет
`parse_number`	логическое значение	Интерпретировать строковые представления чисел (`"1"`) как числовые типы.	`true`	нет

Ошибки

Функция `rv_parse_html` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректным документом HTML.

Примеры

С указанием text_key, без разбора числовых типов

Пример 452. Исходный код

```
value = s'<book category="PROGRAMMING"> <title lang="en">The Rust programming
language</title> <author>Steve Klabnik, Carol Nichols</author>
<year>2019</year></book>'
rv_parse_html!(value,
text_key: "value",
parse_number: false)
```

Пример 453. Результат

```
{
  "book": {
    "@category": "PROGRAMMING",
    "author": "Steve Klabnik, Carol Nichols",
    "title": {
      "@lang": "en",
      "value": "The Rust programming language"
    },
    "year": "2019"
  }
}
```

С использованием text_key во всех случаях

Пример 454. Исходный код

```
value = s'<b>test</b>'
rv_parse_html!(value,
```

```
text_key: "value",
text_key: "node",
always_use_text_key: true)
```

Пример 455. Результат

```
{
  "b": {
    "node": "test"
  }
}
```

Без удаления пробелов между элементами

Пример 456. Исходный код

```
value = s'<root>  <a>test</a>  </root>'
rv_parse_html!(value, trim: false)
```

Пример 457. Результат

```
{
  "root": {
    "a": "test",
    "text": [
      "  ",
      "  "
    ]
  }
}
```

parse_json

Разбирает `value` как JSON.

Чтобы извлечь содержимое из поврежденного JSON, используйте функцию `rv_json_extract_payload`.

Спецификация функции

```
parse_json(value: <строка>, [max_depth: <целое число>, lossy: <логическое значение>])
:: <логическое значение | целое число | число с плавающей точкой | строка | объект |
массив | пустое значение> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка с представлением JSON для разбора.		да
`max_depth`	целое число	Количество уровней для разбора вложенных JSON-документов. Значение должно быть в диапазоне от 1 до 128.		нет
`lossy`	логическое значение	Разбирать ли JSON способом с потерями. Если аргумент установлен в `true`, невалидные UTF-8 символы заменяются на символ Юникода `&#xFFFD;` (U+FFFD), иначе возвращается ошибка, если есть любые невалидные UTF-8 символы.	`true`	нет

Примечания

У этой функции есть специальное поведение, которое следует учитывать:

- Возвращаются только типы JSON. Если нужно преобразовать `string` в `timestamp`, используйте функцию `parse_timestamp`.

Ошибки

Функция `parse_json` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является допустимым JSON-представлением.

Примеры

Разбор JSON

Пример 458. Исходный код

```
parse_json!("{\"key\": \"val\"}")
```

Пример 459. Результат

```
{
  "key": "val"
}
```

Разбор JSON с ограничением глубины

Пример 460. Исходный код

```
parse_json!("{\"top_level\":{\"key\": \"val\"}}",
max_depth: 1)
```

Пример 461. Результат

```
{
  "top_level": "{\"key\": \"val\"}"
}
```

rv_json_extract_payload

Извлекает значение ключа `payload_key` из поврежденного или невалидного JSON.

Спецификация функции

```
rv_json_extract_payload(value: <строка>, payload_key: <строка>)
:: <логическое значение | целое число | число с плавающей точкой | строка | объект |
массив | пустое значение>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка с представлением JSON для извлечения		да
`payload_key`	строка	Ключ, значение которого нужно извлечь		да

Примечания

Особенность функции, о которой нужно знать:

- Если указанный ключ встречается несколько раз, то будет извлечено значение только первого вхождения.

Ошибки

Функция `rv_json_extract_payload` **может возвращать ошибки**, для которых требуется обработка:

- `value` не содержит ключ `payload_key`.

Примеры

JSON с валидной структурой:

Пример 462. Исходный код

```
rv_json_extract_payload(s'{"a": 123, "payload"    : "data"}',
payload_key: "payload")
```

Пример 463. Результат

```
"data"
```

JSON с невалидной структурой:

Пример 464. Исходный код

```
rv_json_extract_payload(s'"asd":123,"payload" : "text"',
payload_key: "payload")
```

Пример 465. Результат

```
"text"
```

Несколько ключей `payload_key`:

Пример 466. Исходный код

```
rv_json_extract_payload(s'{ asd":123,"payload" : "first", "payload": "second" }',
payload_key: "payload")
```

Пример 467. Результат

```
"first"
```

parse_key_value

Разбирает `value` в формате "ключ — значение", также известном как logfmt.

- Ключи и значения могут быть заключены в кавычки `"`.
- Символы `"` могут быть экранированы с помощью обратной косой черты `\`.

Спецификация функции

```
parse_key_value(value: <строка>, [key_value_delimiter: <строка>, field_delimiter:
<строка>, whitespace: <строка>, accept_standalone_key: <логическое значение>])
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да
`key_value_delimiter`	строка	Строка, разделяющая ключ и значение.	`=`	нет
`field_delimiter`	строка	Строка, разделяющая каждую пару "ключ — значение".	(пробел)	нет

Аргумент	Тип	Описание	По умолчанию	Обязателен
`whitespace`	строка	Обработка пробелов вокруг разделителя, заданного в параметре `key_value_delimiter`: • `lenient` — игнорировать пробелы; • `strict` — интерпретировать пробелы как обычные символы.	`lenient`	нет
`accept_standalone_key`	логическое значение	Разрешены ли ключи без значений. Если такие ключи разрешены, в возвращаемом объекте им будет присвоено значение `true`.	`true`	нет

Примечания

У этой функции есть специальное поведение, которое следует учитывать:

- Все значения возвращаются как строки или как массивы строк для дублирующихся ключей. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.

Ошибки

Функция `parse_key_value` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является правильно отформатированной строкой "ключ — значение".

Примеры

Разбор лога в формате logfmt

Пример 468. Исходный код

```
parse_key_value!(
"@timestamp=\"Sun Jan 10 16:47:39 EST 2021\" level=info msg=\"Stopping all fetchers\"
tag#production=stopping_fetchers id=ConsumerFetcherManager-1382721708341
module=kafka.consumer.ConsumerFetcherManager"
)
```

Пример 469. Результат

```
{
  "@timestamp": "Sun Jan 10 16:47:39 EST 2021",
  "level": "info",
  "msg": "Stopping all fetchers",
  "tag#production": "stopping_fetchers",
  "id": "ConsumerFetcherManager-1382721708341",
  "module": "kafka.consumer.ConsumerFetcherManager"
```

```
}
```

Разбор лога с разделителем запятой

Пример 470. Исходный код

```
parse_key_value!(
"path:\"/cart_link\", host:store.app.com, fwd: \"203.0.113.16\", dyno: web.1,
connect:0ms, service:87ms, status:304, bytes:632, protocol:https",
field_delimiter: ",",
key_value_delimiter: ":"
)
```

Пример 471. Результат

```
{
  "path": "/cart_link",
  "host": "store.app.com",
  "fwd": "203.0.113.16",
  "dyno": "web.1",
  "connect": "0ms",
  "service": "87ms",
  "status": "304",
  "bytes": "632",
  "protocol": "https"
}
```

Разбор журнала с ключами без значений

Пример 472. Исходный код

```
parse_key_value!(
"env:prod,service:backend,region:eu-east1,beta",
field_delimiter: ",",
key_value_delimiter: ":",
)
```

Пример 473. Результат

```
{
  "env": "prod",
  "service": "backend",
  "region": "eu-east1",
  "beta": true
}
```

Разбор дублирующихся ключей

Пример 474. Исходный код

```
parse_key_value!(
"at=info,method=GET,path=\"/index\",status=200,tags=dev,tags=dummy",
field_delimiter: ",",
key_value_delimiter: "=",
)
```

Пример 475. Результат

```
{
  "at": "info",
  "method": "GET",
  "path": "/index",
  "status": "200",
  "tags": [
    "dev",
    "dummy"
  ]
}
```

parse_ruby_hash

Разбирает `value` как хэш Ruby.

Спецификация функции

```
parse_ruby_hash(value: <строка>)
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление хэша Ruby для разбора		да

Примечания

Эта функция имеет особое поведение, о котором следует знать.

- Возвращаются только типы Ruby. Если нужно преобразовать `строку` в `метку времени`, используйте функцию `parse_timestamp`.

Ошибки

Функция `parse_ruby_hash` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является допустимым представлением хэша Ruby.

Примеры

Разбор хэша Ruby

Пример 476. Исходный код

```
parse_ruby_hash!(s'{ "test" => "value", "testNum" => 0.2, "testObj" => { "testBool" =>
true, "testNull" => nil } }')
```

Пример 477. Результат

```
{
  "test": "value",
  "testNum": 0.2,
  "testObj": {
    "testBool": true,
    "testNull": null
  }
}
```

parse_xml

Разбирает `value` как XML.

Спецификация функции

```
parse_xml(value: <строка>, [include_attr: <логическое значение>, attr_prefix:
<строка>, text_key: <строка>, always_use_text_key: <логическое значение>, parse_bool:
<логическое значение>, parse_null: <логическое значение>, parse_number: <логическое
значение>])
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строковое представление XML-документа для разбора.		да
`include_attr`	логическое значение	Включать атрибуты XML-тегов в возвращаемый объект.	`true`	нет
`attr_prefix`	строка	Префикс строки для ключей атрибутов XML-тегов.	`@`	нет
`text_key`	строка	Имя ключа для расширенных текстовых узлов.	`text`	нет
`always_use_text_key`	логическое значение	Всегда возвращать узлы текста как `{"<text_key>": "value"}`.	`false`	нет
`parse_bool`	логическое значение	Разбирать "true" и "false" как логические значения.	`true`	нет

Аргумент	Тип	Описание	По умолчанию	Обязателен
`parse_null`	логическое значение	Разбирать "null" как `null`.	`true`	нет
`parse_number`	логическое значение	Разбирать числа как целые/вещественные.	`true`	нет

Примечания

Эта функция имеет особое поведение, о котором следует знать.

- Допустимый XML должен содержать ровно один корневой узел. Всегда возвращает объект.

Ошибки

Функция `parse_xml` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является правильным XML-документом

Примеры

Разбор XML

Пример 478. Исходный код

```
value = s'<book category="CHILDREN"><title lang="en">Harry Potter</title><author>J K.
Rowling</author><year>2005</year></book>'

parse_xml!(value, text_key: "value", parse_number: false)
```

Пример 479. Результат

```
{
  "book": {
    "@category": "CHILDREN",
    "author": "J K. Rowling",
    "title": {
      "@lang": "en",
      "value": "Harry Potter"
    },
    "year": "2005"
  }
}
```

Функции разбора метаданных и параметров

parse_bytes

Разбирает `value` как количество информации в указанных единицах измерения.

Спецификация функции

```
parse_bytes(value: <строка>, unit: <строка>, [base: <строка>])
:: <число с плавающей точкой>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Количество информации в двоичных единицах измерения либо единицах СИ.		да
`unit`	строка	Единицы, в которые нужно перевести объем.		да
`base`	строка	Система приставок, используемая при неоднозначных обозначениях: • "2" — двоичная; • "10" — десятичная.	`"2"`	нет

Используемые обозначения представлены в теблице:

Обозначение	Десятичные единицы	Двоичные единицы
`B`	байт	
`kB`	килобайт	
`KiB`		кибибайт
`MB`	мегабайт	мебибайт (`MiB`)
`GB`	гигабайт	гибибайт (`GiB`)
`TB`	терабайт	тебибайт (`TiB`)
`PB`	петабайт	пебибайт (`PiB`)
`EB`	эксабайт	эксбибайт (`EiB`)

Ошибки

Функция `parse_bytes` **возвращает ошибку**, если `value` не является объемом информации в правильном формате.

Примеры

Перевод кибибайтов в число мебибайт

Пример 480. Исходный код

```
parse_bytes!("1024KiB", unit: "MiB")
```

Пример 481. Результат

```
1
```

Перевод терабайтов в число мегабайт

Пример 482. Исходный код

```
parse_bytes!("4TB", unit: "MB", base: "10")
```

```
4000000
```

Перевод ГБ в число байт с указанием используемой системы

Пример 483. Исходный код

```
parse_bytes!("1GB", unit: "B", base: "2")
```

Пример 484. Результат

```
1073741824
```

parse_duration

Разбирает `value` в удобочитаемый формат длительности, используя единицы, указанные в параметре `unit`.

Спецификация функции

```
parse_duration(value: <строка>, unit: <строка>)
:: <число с плавающей точкой>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка с длительностью. Должна состоять из пар чисел и единиц измерения.		да
`unit`	строка	Единица измерения для вывода длительности.		да

Используемые единицы измерения:

- `ns` — наносекунды;
- `us`, `µs` — микросекунды;
- `ms` — миллисекунды;

- `cs` — сантисекунды;
- `ds` — децисекунды;
- `s` — секунды;
- `m` — минуты;
- `h` — часы;
- `d` — сутки.

Ошибки

Функция `parse_duration` **может возвращать ошибки**, для которых требуется обработка:

- `value` не имеет правильного формата длительности

Примеры

Преобразование миллисекунд в секунды

Пример 485. Исходный код

```
parse_duration!("1005ms", unit: "s")
```

Пример 486. Результат

```
1.005
```

Преобразование секунд и минут в миллисекунды

Пример 487. Исходный код

```
parse_duration!("1s1m", unit: "ms")
```

Пример 488. Результат

```
61000
```

parse_query_string

Разбирает `value` как строку запроса (query string).

Спецификация функции

```
parse_query_string(value: <строка>)
:: <объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да

Примечания

Эта функция имеет особое поведение, о котором следует знать.

- Все значения возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.
- Пустые ключи и значения разрешены.

Примеры

Разбор строки запроса

Пример 489. Исходный код

```
parse_query_string("foo=%2B1&bar=2&bar=3&xyz")
```

Пример 490. Результат

```
{
  "bar": [
    "2",
    "3"
  ],
  "foo": "+1",
  "xyz": ""
}
```

Разбор строки запроса Ruby on Rails

Пример 491. Исходный код

```
parse_query_string("?foo%5b%5d=1&foo%5b%5d=2")
```

Пример 492. Результат

```
{
  "foo[]": [
    "1",
    "2"
  ]
}
```

parse_tokens

Разбирает `value` в формате "token". Токенами считаются такие последовательности символов:

- Слова, окруженные пробелами.
- Тексты, заключенные в двойные кавычки: `".."`. Кавычки можно включить в токен, если они экранированы обратной косой чертой (\).
- Тексты, заключенные в квадратные скобки: `[..]`. Закрывающие квадратные скобки можно включить в токен, если они экранированы обратной косой чертой (\).

Спецификация функции

```
parse_tokens(value: <строка>)
:: <массив> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для токенизации.		да

Примечания

Эта функция имеет особое поведение, о котором следует знать.

- Все значения токенов возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.

Ошибки

Функция `parse_tokens` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является правильно отформатированной строкой с токенами.

Примеры

Разбор токенов

Пример 493. Исходный код

```
parse_tokens(
"A sentence \"with \\\"a\\\" sentence inside\" and [some brackets]"
)
```

Пример 494. Результат

```
[
  "A",
  "sentence",
  "with \\\"a\\\" sentence inside",
  "and",
```

```
  "some brackets"
]
```

parse_url

Разбирает `value` в формате URL.

Спецификация функции

```
parse_url(value: <строка>, [default_known_ports: <логическое значение>])
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Текст URL.		да
`default_known_ports`	логическое значение	Заполнять ли номер порта значением по умолчанию. Если во входном URL не указан номер порта или он совпадает со стандартным, поле будет заполнено значением по умолчанию. Это применимо к протоколам `http`, `https`, `ftp`, `ws` и `wss`.	`false`	нет

Примечания

Особенности функции, о которых нужно знать:

- По умолчанию, если во входном URL указан стандартный порт, он не учитывается. Поле `port` в возвращаемом объекте остается пустым.

Ошибки

Функция `parse_url` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является правильно отформатированным URL.

Примеры

Разбор URL

Пример 495. Исходный код

```
parse_url!("ftp://foo:bar@example.com:4343/foobar?hello=world#123")
```

Пример 496. Результат

```
{
  "scheme": "ftp",
  "username": "foo",
  "password": "bar",
```

```
  "host": "example.com",
  "port": 4343,
  "path": "/foobar",
  "query": {
    "hello": "world"
  },
  "fragment": "123"
}
```

Разбор URL с портом по умолчанию

Пример 497. Исходный код

```
parse_url!("https://example.com", default_known_ports: true)
```

Пример 498. Результат

```
{
  "scheme": "https",
  "username": "",
  "password": "",
  "host": "example.com",
  "port": 443,
  "path": "/",
  "query": {},
  "fragment": null
}
```

parse_user_agent

Разбирает `value` как строку пользовательского агента. Формат пользовательского агента имеет нечеткое определение, поэтому функция обеспечивает лишь максимально возможные результаты, а не точный разбор.

Спецификация функции

```
parse_user_agent(value: <строка>, [mode: <строка>])
:: <объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`mode`	строка	Определяет характеристики производительности и надежности: • `fast` — самый быстрый, но менее надежный режим. Используется синтаксический анализатор из проекта Woothee. • `reliable` — обеспечивает большую надежность, чем режим `fast`, и в большинстве случаев не снижает скорость работы. Используется синтаксический анализатор Wood These. Если Wood These не разобрал какие-либо поля, используется анализатор uapproject. • `enriched` — объединяет результаты Wood These и uapproject.	`fast`	нет

Примечания

Эта функция имеет особое поведение, о котором следует знать:

- Все значения возвращаются как строки или как `null`. Все значения возвращаются как строки. Рекомендуется преобразовывать значения в нужные типы, используя соответствующие функции.
- Разные режимы возвращают разные схемы.
- Поля, которые не были разобраны, устанавливаются как `null`.

Примеры

Быстрый режим

Пример 499. Исходный код

```
parse_user_agent(
"Mozilla Firefox 1.0.1 Mozilla/5.0 (X11; U; Linux i686; de-DE; rv:1.7.6)
Gecko/20050223 Firefox/1.0.1"
)
```

Пример 500. Результат

```
{
  "browser": {
    "family": "Firefox",
    "version": "1.0.1"
  },
```

```
  "device": {
    "category": "pc"
  },
  "os": {
    "family": "Linux",
    "version": null
  }
}
```

Надежный режим

Пример 501. Исходный код

```
parse_user_agent(
"Mozilla/4.0 (compatible; MSIE 7.66; Windows NT 5.1; SV1; .NET CLR 1.1.4322)",
mode: "reliable"
)
```

Пример 502. Результат

```
{
  "browser": {
    "family": "Internet Explorer",
    "version": "7.66"
  },
  "device": {
    "category": "pc"
  },
  "os": {
    "family": "Windows XP",
    "version": "NT 5.1"
  }
}
```

Расширенный режим

Пример 503. Исходный код

```
parse_user_agent(
"Opera/9.80 (J2ME/MIDP; Opera Mini/4.3.24214; iPhone; CPU iPhone OS 4_2_1 like Mac OS
X; AppleWebKit/24.783; U; en) Presto/2.5.25 Version/10.54",
mode: "enriched"
)
```

Пример 504. Результат

```
{
  "browser": {
    "family": "Opera Mini",
```

```
    "major": "4",
    "minor": "3",
    "patch": "24214",
    "version": "10.54"
  },
  "device": {
    "brand": "Apple",
    "category": "smartphone",
    "family": "iPhone",
    "model": "iPhone"
  },
  "os": {
    "family": "iOS",
    "major": "4",
    "minor": "2",
    "patch": "1",
    "patch_minor": null,
    "version": "4.2.1"
  }
}
```

Функции преобразования IP

ip_aton

Преобразует IPv4-адрес в представлении чисел и точек в байты, представленные в сетевом порядке, как целое число.

Функция повторяет поведение функции inet_aton.

Спецификация функции

```
ip_aton(value: <строка>)
:: <целое число> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
value	строка	IP-адрес для преобразования в бинарный формат.		да

Ошибки

Функция ip_aton **может возвращать ошибки**, для которых требуется обработка:

- value не является допустимым IPv4-адресом.

Примеры

IPv4 в целое число

Пример 505. Исходный код

```
ip_aton!("233.252.0.4")
```

Пример 506. Результат

```
3925606404
```

ip_cidr_contains

Определяет, содержится ли `ip` в блоке, заданном с помощью `cidr`.

Спецификация функции

```
ip_cidr_contains(cidr: <строка | массив>, ip: <строка>)
:: <логическое значение>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`cidr`	строка массив	Маска CIDR (v4 или v6).		да
`ip`	строка	IP-адрес (v4 или v6).		да

Ошибки

Функция `ip_cidr_contains` **может возвращать ошибки**, для которых требуется обработка:

- `cidr` не является допустимой маской CIDR.
- `ip` не является допустимым IP-адресом.

Примеры

IPv4 содержит CIDR

Пример 507. Исходный код

```
ip_cidr_contains!("203.0.113.0/24", "203.0.113.32")
```

Пример 508. Результат

```
true
```

IPv4 является частным

Пример 509. Исходный код

```
ip_cidr_contains!(["10.0.0.0/8", "172.16.0.0/12", "192.168.0.0/16"], "192.168.10.32")
```

Пример 510. Результат

```
true
```

Пример 511. Исходный код

```
ip_cidr_contains!("2001:db8:4:ba::/64", "2001:db8:4:ba:2e0:81ff:fe22:d1f1")
```

Пример 512. Результат

```
true
```

ip_ntoa

Преобразует числовое представление IPv4-адреса в байтах, представленных в сетевом порядке, в представление чисел и точек.

Функция повторяет поведение функции inet_ntoa.

Спецификация функции

```
ip_ntoa(value: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Целочисленное представление IPv4-адреса.		да

Ошибки

Функция `ip_ntoa` **может возвращать ошибки**, для которых требуется обработка:

- `value` не может быть представлено в u32.

Примеры

Целое число в IPv4

Пример 513. Исходный код

```
ip_ntoa!(3925606404)
```

Пример 514. Результат

```
233.252.0.4
```

ip_ntop

Преобразует IPv4 и IPv6-адреса из бинарной формы в текстовую форму.

Функция повторяет поведение функции inet_ntop.

Спецификация функции

```
ip_ntop(value: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Двоичные данные для преобразования. Для IPv4-адресов длина должна быть 4 байта (32 бита). Для IPv6-адресов длина должна быть 16 байт (128 бит).		да

Примечания

Эта функция имеет особое поведение, о котором стоит знать.

- Двоичные данные для этой функции не всегда легко отобразить в печатном виде. Однако результаты функций, таких как `decode_base64` или `decode_percent`, все равно можно использовать корректно.

Ошибки

Функция `ip_ntop` **может возвращать ошибки**, для которых требуется обработка:

- `value` должно быть длиной 4 или 16 байт

Примеры

Преобразование IPv4-адреса из байтов после декодирования из Base64

Пример 515. Исходный код

```
ip_ntop!(decode_base64!("wKgAAQ=="))
```

Пример 516. Результат

```
192.168.0.1
```

Преобразование IPv6-адреса из байтов после декодирования из Base64

Пример 517. Исходный код

```
ip_ntop!(decode_base64!("IAENuIWjAAAAAIouA3BzNA=="))
```

Пример 518. Результат

```
2001:db8:85a3::8a2e:370:7334
```

ip_pton

Преобразует IPv4 и IPv6-адреса из текстовой формы в бинарную форму.

- Бинарная форма IPv4-адресов имеет длину 4 байта (32 бита).
- Бинарная форма IPv6-адресов имеет длину 16 байт (128 бит).

Функция повторяет поведение функции inet_pton.

Спецификация функции

```
ip_pton(value: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	IP-адрес (v4 или v6) для преобразования в бинарную форму.		да

Примечания

Эта функция имеет особое поведение, о котором стоит знать.

- Двоичные данные из этой функции не всегда могут быть удобочитаемыми. Однако результаты функций, таких как `encode_base64` или `encode_percent`, все равно можно корректно обрабатывать.

Ошибки

Функция `ip_pton` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является допустимым IP-адресом (v4 или v6) в текстовой форме.

Примеры

Преобразование IPv4-адреса в байты и кодирование в Base64

Пример 519. Исходный код

```
encode_base64(ip_pton!("192.168.0.1"))
```

Пример 520. Результат

```
wKgAAQ==
```

Преобразование IPv6-адреса в байты и кодирование в Base64

Пример 521. Исходный код

```
encode_base64(ip_pton!("2001:db8:85a3::8a2e:370:7334"))
```

Пример 522. Результат

```
IAENuIWjAAAAAIouA3BzNA==
```

ip_subnet

Извлекает подсетевой адрес из `ip` с использованием заданной подсети (`subnet`).

Спецификация функции

```
ip_subnet(ip: <строка>, subnet: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`ip`	строка	IP-адрес (v4 или v6).		да
`subnet`	строка	Подсеть, которую необходимо извлечь из IP-адреса. Это может быть либо длина префикса, например, `/8`, либо сетевая маска, например, `255.255.0.0`. Сетевая маска может быть как IPv4, так и IPv6 адресом.		да

Примечания

Эта функция имеет особое поведение, о котором стоит знать.

- Работает как с IPv4, так и с IPv6 адресами. Версия IP для маски должна быть такой же, как у предоставленного адреса.

Ошибки

Функция `ip_subnet` **может возвращать ошибки**, для которых требуется обработка:

- `ip` не является допустимым IP-адресом.
- `subnet` не является допустимой подсетью.

Примеры

Подсеть IPv4

Пример 523. Исходный код

```
ip_subnet!("192.168.10.32", "255.255.255.0")
```

Пример 524. Результат

```
192.168.10.0
```

Подсеть IPv6

Пример 525. Исходный код

```
ip_subnet!("2404:6800:4003:c02::64", "/32")
```

Пример 526. Результат

```
2404:6800::
```

ip_to_ipv6

Преобразует `ip` в адрес IPv6.

Спецификация функции

```
ip_to_ipv6(ip: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`ip`	строка	IP-адрес, который необходимо преобразовать в IPv6.		да

Ошибки

Функция `ip_to_ipv6` **может возвращать ошибки**, для которых требуется обработка:

- `ip` не является допустимым IP-адресом.

Примеры

IPv4 в IPv6

Пример 527. Исходный код

```
ip_to_ipv6!("192.168.10.32")
```

Пример 528. Результат

```
::ffff:192.168.10.32
```

ipv6_to_ipv4

Преобразует `ip` в адрес IPv4. Если `ip` уже является IPv4-адресом, то он возвращается без изменений. Если `ip` в данный момент является IPv6-адресом, то он должен быть совместим с IPv4, в противном случае возникнет ошибка.

Спецификация функции

```
ipv6_to_ipv4(ip: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`ip`	строка	IPv4-совместимый IPv6-адрес для преобразования.		да

Ошибки

Функция `ipv6_to_ipv4` **может возвращать ошибки**, для которых требуется обработка:

- `ip` не является допустимым IP-адресом.
- `ip` является IPv6-адресом, который не совместим с IPv4.

Примеры

IPv6 в IPv4

Пример 529. Исходный код

```
ipv6_to_ipv4!("::ffff:192.168.0.1")
```

Пример 530. Результат

```
192.168.0.1
```

Функции кодирования и декодирования

В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции декодирования

decode_base16

Декодирует `value` (строку в формате Base16) в исходную строку.

Спецификация функции

```
decode_base16(value: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Данные в формате Base16 для декодирования.		да

Ошибки

Функция `decode_base16` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректной закодированной строкой в формате Base16.

Примеры

Декодирование данных в формате Base16

Пример 531. Исходный код

```
decode_base16!("796f7520686176652073 756363657373667 56c6c79206465636f646564206d65")
```

Пример 532. Результат

```
you have successfully decoded me
```

decode_base64

Декодирует `value` (строка в формате Base64) в исходную строку.

Спецификация функции

```
decode_base64(value: <строка>, [charset: <строка>])
```

```
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Данные в формате Base64 для декодирования.		да
`charset`	строка	Набор символов, используемый при декодировании данных: • `standard` — стандартный формат Base64; • `url_safe` — измененный Base64 для URL.	`standard`	нет

Ошибки

Функция `decode_base64` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректной закодированной строкой в формате Base64.

Примеры

Декодирование данных в формате Base64 (по умолчанию)

Пример 533. Исходный код

```
decode_base64!("eW91IGhhdmUgc3VjY2Vzc2Z1bGx5IGRlY29kZWQgbWU=")
```

Пример 534. Результат

```
you can't make your heart feel something it won't
```

Декодирование данных в измененном формате Base64 для URL

Пример 535. Исходный код

```
decode_base64!("eW91IGNhbid0IG1ha2UgeW91ciBoZWFydCBmZWVsIHNvbWV0aGluZyBpdCB3b24ndA==",
charset: "url_safe")
```

Пример 536. Результат

```
you can't make your heart feel something it won't
```

decode_charset

Преобразует строку, закодированную не в UTF-8, в строку UTF-8, используя заданную кодировку.

Спецификация функции

```
decode_charset(value: <строка>, from_charset: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка не в UTF-8, которую нужно декодировать.		да
`from_charset`	строка	Набор символов, используемый для декодирования. Допустимые значения см. в списке.		да

Ошибки

Функция `decode_charset` **возвращает ошибки**, для которых требуется обработка, если `from_charset` не является корректным названием кодировки.

Примеры

Декодирование строки в кодировке EUC-KR

Пример 537. Исходный код

```
decode_charset!(decode_base64!("vsiz58fPvLy/5A=="), "euc-kr")
```

Пример 538. Результат

```
□□□□□
```

Декодирование строки в кодировке EUC-JP

Пример 539. Исходный код

```
decode_charset!(decode_base64!("pLOk86TLpMGkzw=="), "euc-jp")
```

Пример 540. Результат

```
こんにちは
```

Декодирование строки в кодировке GB2312

Пример 541. Исходный код

```
decode_charset!(decode_base64!("xOO6ww=="), "gb2312")
```

Пример 542. Результат

```
好
```

decode_gzip

Декодирует `value` (строка в формате Gzip) в исходную строку.

Спецификация функции

```
decode_gzip(value: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Данные в формате Gzip для декодирования.		да

Ошибки

Функция `decode_gzip` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректной строкой, закодированной в формате Gzip.

Примеры

Декодирование данных в формате Gzip

Пример 543. Исходный код

```
encoded_text =
decode_base64!("H4sIAHEAymMAA6vML1XISCxLVSguTU5OLS5OK83JqVRISU3OT0lNUchNBQD7BGDaIAAAAA
==")
decode_gzip!(encoded_text)
```

Пример 544. Результат

```
you have successfully decoded me
```

rv_decode_lz4

Декодирует `value` как запись, сжатую с помощью алгоритма LZ4.

Спецификация функции

```
rv_decode_lz4(value: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Запись, сжатая алгоритмом LZ4.		да

Ошибки

Функция `rv_decode_lz4` **может возвращать ошибки**, для которых требуется обработка:

- Не удается декодировать запись как LZ4.

Пример

Пример 545. Исходный код

```
rv_decode_lz4!(decode_base64!("CQAAAJBlbmNvZGVfbWU="))
```

Пример 546. Результат

```
encode_me
```

decode_mime_q

Заменяет подстроки закодированных слов в кодировке q или base64 в `value` их исходной строкой.

Спецификация функции

```
decode_mime_q(value: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Закодированная строка с кодировочным словом для декодирования.		да

Ошибки

Функция `decode_mime_q` **может возвращать ошибки**, для которых требуется обработка:

- `value` содержит некорректную закодированную строку с кодировочными словами.

Примеры

Декодирование одиночной с кодировочными словами

Пример 547. Исходный код

```
decode_mime_q!("=?utf-8?b?SGVsbG8sIFdvcmxkIQ==?=")
```

Пример 548. Результат

```
Hello, World!
```

Встроенная

Пример 549. Исходный код

```
decode_mime_q!("From: =?utf-8?b?SGVsbG8sIFdvcmxkIQ==?= <=?utf-
8?q?hello=5Fworld=40example=2ecom?=>")
```

Пример 550. Результат

```
From: Hello, World! <hello_world@example.com>
```

Без указания набора символов

Пример 551. Исходный код

```
decode_mime_q!("?b?SGVsbG8sIFdvcmxkIQ==")
```

Пример 552. Результат

```
Hello, World!
```

decode_percent

Декодирует строку `value`, закодированную в формате percent, как URL.

Спецификация функции

```
decode_percent(value: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для декодирования.		да

Примеры

Процентное декодирование значения

Пример 553. Исходный код

```
decode_percent("foo%20bar%3F")
```

Пример 554. Результат

```
foo bar?
```

decode_punycode

Декодирует строку `value`, закодированную в формате punycode, как IDN (интернационализированное доменное имя).

Спецификация функции

```
decode_punycode(value: <строка>, [validate: <логическое значение>])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для декодирования.		да
`validate`	логическое значение	Если `true`, проверяет, является ли входная строка валидным именем домена.	`true`	нет

Ошибки

Функция `decode_punycode` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректной строкой punycode.

Примеры

Декодирование интернационализированного доменного имени, закодированного в punycode

Пример 555. Исходный код

```
decode_punycode!("xn--e1afmkfd.xn--80akhbyknj4f")
```

Пример 556. Результат

```
пример.испытание
```

Декодирование строки из символов ASCII

Пример 557. Исходный код

```
decode_punycode!("example.com")
```

Пример 558. Результат

```
example.com
```

Декодирование без валидации

Пример 559. Исходный код

```
decode_punycode!("xn--8hbb.xn--fiba.xn--8hbf.xn--eib.", validate: false)
```

Пример 560. Результат

```
□□.□□.□□.□.
```

decode_snappy

Декодирует `value`, представленное в формате Snappy, в исходную строку.

Спецификация функции

```
decode_snappy(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Данные Snappy для декодирования.		да

Ошибки

Функция `decode_snappy` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректной строкой, закодированной в формате Snappy.

Пример

Пример 561. Исходный код

```
encoded_text = decode_base64!("G2hBIHVzZXIgYWNjb3VudCB3YXMgY3JlYXRlZC4=")
decode_snappy!(encoded_text)
```

Пример 562. Результат

```
"A user account was created."
```

decode_zlib

Декодирует `value` (строка в формате Zlib) в исходную строку.

Спецификация функции

```
decode_zlib(value: <строка>)
```

```
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Данные в формате Zlib для декодирования.		да

Ошибки

Функция `decode_zlib` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректной закодированной строкой в формате Zlib.

Примеры

Декодирование данных в формате Zlib

Пример 563. Исходный код

```
encoded_text =
decode_base64!("eJwNy4ENwCAIBMCNXIlQ/KqplUSgCdvXAS41qPMHshCB2R1zJlWIVlR6UURX2+wx2YcuK3
kAb9C1wd6dn7Fa+QH9gRxr")
decode_zlib!(encoded_text)
```

Пример 564. Результат

```
you have successfully decoded me
```

decode_zstd

Декодирует `value` (строка в формате Zstandard) в исходную строку.

Спецификация функции

```
decode_zstd(value: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Данные в формате Zstandard для декодирования.		да

Ошибки

Функция `decode_zstd` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является корректной закодированной строкой в формате Zstd.

Примеры

Декодирование данных в формате Zstd

Пример 565. Исходный код

```
encoded_text =
decode_base64!("KLUv/QBY/QEAYsQOFKClbQBedqXsb96EWDax/f/F/z+gNU4ZTInaUeAj82KqPFjUzKqhcf
DqAIsLvAsnY1bI/N2mHzDixRQA")
decode_zstd!(encoded_text)
```

Пример 566. Результат

```
you have successfully decoded me
```

Функции кодирования

encode_base16

Кодирует `value` в Base16.

Спецификация функции

```
encode_base16(value: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо закодировать.		да

Примеры

Кодирование в Base16

Пример 567. Исходный код

```
encode_base16("please encode me")
```

Пример 568. Результат

```
706c6561736520656e636f6465206d65
```

encode_base64

Кодирует `value` в Base64.

Спецификация функции

```
encode_base64(value: <строка>, [padding: <логическое значение>, charset: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо закодировать.		да
`padding`	логическое значение	Дополняется ли вывод Base64.	`true`	нет
`charset`	строка	Набор символов, используемый при кодировании данных: • `standard` — стандартный формат Base64; • `url_safe` — измененный Base64 для URL.	`standard`	нет

Примеры

Кодирование в Base64 (по умолчанию)

Пример 569. Исходный код

```
encode_base64("please encode me")
```

Пример 570. Результат

```
cGxlYXNlIGVuY29kZSBtZQ==
```

Кодирование в Base64 (без дополнения)

Пример 571. Исходный код

```
encode_base64("please encode me, no padding though", padding: false)
```

Пример 572. Результат

```
cGxlYXNlIGVuY29kZSBtZSwgbm8gcGFkZGluZyB0aG91Z2g
```

Кодирование в измененный Base64 для URL

Пример 573. Исходный код

```
encode_base64("please encode me, but safe for URLs", charset: "url_safe")
```

Пример 574. Результат

```
cGxlYXNlIGVuY29kZSBtZSwgYnV0IHNhZmUgZm9yIFVSTHM=
```

encode_charset

Преобразует строку, закодированную в UTF-8, в строку заданной кодировки.

Спецификация функции

```
encode_charset(value: <строка>, to_charset: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка в UTF-8, которую нужно кодировать.		да
`to_charset`	строка	Набор символов, используемый для кодирования. Допустимые значения см. в списке.		да

Ошибки

Функция `encode_charset` **возвращает ошибки**, для которых требуется обработка, если `to_charset` не является корректным названием кодировки.

Примеры

Кодирование строки UTF-8 в EUC-KR

Пример 575. Исходный код

```
encode_base64(encode_charset!("□□□□□", "euc-kr"))
```

Пример 576. Результат

```
vsiz58fPvLy/5A==
```

Кодирование строки UTF8 в EUC-JP

Пример 577. Исходный код

```
encode_base64(encode_charset!("こんにちは", "euc-jp"))
```

Пример 578. Результат

```
pLOk86TLpMGkzw==
```

Кодирование строки UTF-8 в GB2312

Пример 579. Исходный код

```
encode_base64(encode_charset!("好", "gb2312"))
```

Пример 580. Результат

```
xOO6ww==
```

encode_gzip

Кодирует `value` в формат Gzip.

Спецификация функции

```
encode_gzip(value: <строка>, [compression_level: <целое число>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо закодировать.		да
`compression_level`	целое значение	Уровень сжатия по умолчанию.	`6`	нет

Примеры

Кодирование в Gzip

Пример 581. Исходный код

```
encoded_text = encode_gzip("please encode me")
encode_base64(encoded_text)
```

Пример 582. Результат

```
H4sIAAAAAAAA/yvISU0sTlVIzUvOT0lVyE0FAI4R4vcQAAAA
```

encode_percent

Преобразует `value` в формат с использованием процентного кодирования для безопасного

использования с URL.

Спецификация функции

```
encode_percent(value: <строка>, [ascii_set: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо преобразовать.		да
`ascii_set`	строка	Набор символов ASCII, используемый при преобразовании данных.	`NON_ALPHANUMERIC`	нет

Примеры

Процентное кодирование всех символов, не являющихся буквенно-цифровыми (по умолчанию)

Пример 583. Исходный код

```
encode_percent("foo bar?")
```

Пример 584. Результат

```
foo%20bar%3F
```

Процентное кодирование только управляющих символов

Пример 585. Исходный код

```
encode_percent("foo bar", ascii_set: "CONTROLS")
```

Пример 586. Результат

```
foo %09bar
```

encode_proto

Кодирует `value` в payload — полезную нагрузку Protocol Buffers.

Спецификация функции

```
encode_proto(value: <объект>, desc_file: <строка>, message_type: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Объект для преобразования в payload Protocol Buffers.		да
`desc_file`	строка	Путь к файлу набора дескрипторов Protocol Buffers. Должен быть строковым литерал. Этот файл является результатом работы команды `protoc -o`.		да
`message_type`	строка	Имя типа сообщения, используемого для сериализации. Должно быть строковым литералом.		да

Ошибки

Функция `encode_proto` **может возвращать ошибки**, для которых требуется обработка:

- Файл, указанный в аргументе `desc_file`, не существует.
- Тип сообщения, указанный в аргументе `message_type`, отсутствует в файле дескрипторов.

Пример

Пример 587. Исходный код

```
.payload = encode_base64(encode_proto!({"name": "someone", "phones": [{"number":
"123456"}]}, "resources/protobuf_descriptor_set.desc", "test_protobuf.Person"))
```

Пример 588. Результат

```
Cgdzb21lb25lIggKBjEyMzQ1Ng==
```

encode_punycode

Кодирует `значение` в формат punycode. Полезно для IDN.

Спецификация функции

```
encode_punycode(value: <строка>, [validate: <логическое значение>])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для кодирования.		да
`validate`	логическое значение	Проверять ли, что входная строка является валидным именем домена.	`true`	нет

Ошибки

Функция `encode_punycode` **может возвращать ошибки**, для которых требуется обработка:

- `value` нельзя закодировать в punycode.

Примеры

Кодирование интернационализированного доменного имени

Пример 589. Исходный код

```
encode_punycode!("пример.испытание")
```

Пример 590. Результат

```
xn--e1afmkfd.xn--80akhbyknj4f
```

Кодирование интернационализированного доменного имени в смешанном регистре

Пример 591. Исходный код

```
encode_punycode!("пРИМер.испытание")
```

Пример 592. Результат

```
xn--e1afmkfd.xn--80akhbyknj4f
```

Кодирование строки из символов ASCII

Пример 593. Исходный код

```
encode_punycode!("example.com")
```

Пример 594. Результат

```
example.com
```

Кодирование без валидации

Пример 595. Исходный код

```
encode_punycode!("xn--8hbb.xn--fiba.xn--8hbf.xn--eib.", validate: false)
```

Пример 596. Результат

```
xn--8hbb.xn--fiba.xn--8hbf.xn--eib.
```

Кодирование доменного имени в смешанном регистре без валидации

Пример 597. Исходный код

```
encode_punycode!("www.CAFé.example")
```

Пример 598. Результат

```
"www.xn--caf-dma.example"
```

encode_snappy

Кодирует `value` в формат Snappy.

Спецификация функции

```
encode_snappy(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для кодирования.		да

Ошибки

Функция `encode_snappy` **может возвращать ошибки**, для которых требуется обработка:

- `value` нельзя закодировать в формат Snappy.

Пример

Пример 599. Исходный код

```
encoded_text = encode_snappy!("A user account was created.")
encode_base64(encoded_text)
```

Пример 600. Результат

```
G2hBIHVzZXIgYWNjb3VudCB3YXMgY3JlYXRlZC4=
```

encode_zlib

Кодирует `value` в формат Zlib.

Спецификация функции

```
encode_zlib(value: <строка>, [compression_level: <целое число>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо закодировать.		да
`compression_level`	целое число	Уровень сжатия по умолчанию.	`6`	нет

Примеры

Кодирование в формат Zlib

Пример 601. Исходный код

```
encoded_text = encode_zlib("please encode me")
encode_base64(encoded_text)
```

Пример 602. Результат

```
eJwryElNLE5VSM1Lzk9JVchNBQA0RQX7
```

encode_zstd

Кодирует `value` в формат Zstandard.

Спецификация функции

```
encode_zstd(value: <строка>, [compression_level: <целое число>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо закодировать.		да
`compression_level`	целое число	Уровень сжатия по умолчанию.	`3`	нет

Примеры

Кодирование в формат Zstandard

Пример 603. Исходный код

```
encoded_text = encode_zstd("please encode me")
encode_base64(encoded_text)
```

Пример 604. Результат

```
KLUv/QBYgQAAcGxlYXNlIGVuY29kZSBtZQ==
```

Функции сериализации

encode_json

Кодирует `value` в JSON.

Спецификация функции

```
encode_json(value: <любой тип>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любое	Значение, которое необходимо преобразовать в JSON-строку.		да

Примеры

Кодирование в JSON

Пример 605. Исходный код

```
.payload = encode_json({"hello": "world"})
```

Пример 606. Результат

```
{"hello":"world"}
```

encode_key_value

Кодирует `value` в формат "ключ — значение" с настраиваемыми разделителями. По умолчанию используются разделители, соответствующие формату logfmt.

Спецификация функции

```
encode_key_value(value: <объект>, [fields_ordering: <массив>, key_value_delimiter:
<строка>, field_delimiter: <строка>, flatten_boolean: <логическое значение>])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Значение, которое необходимо преобразовать в строку.		да
`fields_ordering`	массив	Упорядочивание полей для сохранения порядка. Любые поля, отсутствующие в этом списке, будут отображаться неупорядоченными, после всех упорядоченных полей.		нет
`key_value_delimiter`	строка	Строка, разделяющая ключ и значение.	`=`	нет
`field_delimiter`	строка	Строка, разделяющая каждую пару "ключ — значение".	(пробел)	нет
`flatten_boolean`	логическое значение	Указывает, следует ли кодировать "ключ — значение" с логическим значением как отдельный ключ.	`false`	нет

Примечания

У этой функции есть особое поведение, о котором стоит знать:

- Если указан `fields_ordering`, функция может возвращать ошибку, иначе она ошибки не возвращает.

Ошибки

Функция `encode_key_value` **может возвращать ошибки**, для которых требуется обработка:

- В `fields_ordering` содержится элемент, не являющийся строкой.

Примеры

Кодирование с разделителями по умолчанию (без упорядочивания)

Пример 607. Исходный код

```
encode_key_value({"ts": "2021-06-05T17:20:00Z", "msg": "This is a message", "lvl":
"info"})
```

Пример 608. Результат

```
lvl=info msg="This is a message" ts=2021-06-05T17:20:00Z
```

Кодирование с разделителями по умолчанию (с упорядочиванием полей)

Пример 609. Исходный код

```
encode_key_value!({"ts": "2021-06-05T17:20:00Z", "msg": "This is a message", "lvl":
```

```
"info", "log_id": 12345}, ["ts", "lvl", "msg"])
```

Пример 610. Результат

```
ts=2021-06-05T17:20:00Z lvl=info msg="This is a message" log_id=12345
```

Кодирование с разделителями по умолчанию (вложенные поля)

Пример 611. Исходный код

```
encode_key_value({"agent": {"name": "vector"}, "log": {"file": {"path": "my.log"}},
"event": "log"})
```

Пример 612. Результат

```
agent.name=vector event=log log.file.path=my.log
```

Кодирование с разделителями по умолчанию (вложенные поля с упорядочиванием)

Пример 613. Исходный код

```
encode_key_value!({"agent": {"name": "vector"}, "log": {"file": {"path": "my.log"}},
"event": "log"}, ["event", "log.file.path", "agent.name"])
```

Пример 614. Результат

```
event=log log.file.path=my.log agent.name=vector
```

Кодирование с пользовательскими разделителями (без упорядочивания)

Пример 615. Исходный код

```
encode_key_value(
{"ts": "2021-06-05T17:20:00Z", "msg": "This is a message", "lvl": "info"},
field_delimiter: ",",
key_value_delimiter: ":"
)
```

Пример 616. Результат

```
lvl:info,msg:"This is a message",ts:2021-06-05T17:20:00Z
```

Кодирование с пользовательскими разделителями и сглаживанием логических значений

Пример 617. Исходный код

```
encode_key_value(
```

```
{"ts": "2021-06-05T17:20:00Z", "msg": "This is a message", "lvl": "info", "beta":
true, "dropped": false},
field_delimiter: ",",
key_value_delimiter: ":",
flatten_boolean: true
)
```

Пример 618. Результат

```
beta,lvl:info,msg:"This is a message",ts:2021-06-05T17:20:00Z
```

encode_logfmt

Кодирует `value` в формат logfmt.

Спецификация функции

```
encode_logfmt(value: <объект>, [fields_ordering: <массив>])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Значение, которое необходимо преобразовать в строку в формате logfmt.		да
`fields_ordering`	массив	Упорядочивание полей для сохранения порядка. Любые поля, отсутствующие в этом списке, будут отображаться неупорядоченными, после всех упорядоченных полей.		нет

Примечания

У этой функции есть особое поведение, о котором стоит знать:

- Если указан `fields_ordering`, функция может возвращать ошибку, иначе она ошибки не возвращает.

Ошибки

Функция `encode_logfmt` может **возвращать ошибки**, и для их обработки требуется следующее:

- В `fields_ordering` содержится элемент, не являющийся строкой.

Примеры

Кодирование в формат logfmt (без упорядочивания)

Пример 619. Исходный код

```
encode_logfmt({"ts": "2021-06-05T17:20:00Z", "msg": "This is a message", "lvl":
"info"})
```

Пример 620. Результат

```
lvl=info msg="This is a message" ts=2021-06-05T17:20:00Z
```

Кодирование в формат logfmt (с упорядочиванием полей)

Пример 621. Исходный код

```
encode_logfmt!({"ts": "2021-06-05T17:20:00Z", "msg": "This is a message", "lvl":
"info", "log_id": 12345}, ["ts", "lvl", "msg"])
```

Пример 622. Результат

```
ts=2021-06-05T17:20:00Z lvl=info msg="This is a message" log_id=12345
```

Кодирование в формат logfmt (вложенные поля)

Пример 623. Исходный код

```
encode_logfmt({"agent": {"name": "vector"}, "log": {"file": {"path": "my.log"}},
"event": "log"})
```

Пример 624. Результат

```
agent.name=vector event=log log.file.path=my.log
```

Кодирование в формат logfmt (вложенные поля с упорядочиванием)

Пример 625. Исходный код

```
encode_logfmt!({"agent": {"name": "vector"}, "log": {"file": {"path": "my.log"}},
"event": "log"}, ["event", "log.file.path", "agent.name"])
```

Пример 626. Результат

```
event=log log.file.path=my.log agent.name=vector
```

Функции событий и активных списков

В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции секретов

get_secret

Возвращает значение заданного секрета из события.

Спецификация функции

```
get_secret(key: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
key	строка	Имя секрета.		да

Примеры

Получить ключ API Datadog из метаданных события.

Пример 627. Исходный код

```
get_secret("datadog_api_key")
```

Пример 628. Результат

```
secret value
```

remove_secret

Удаляет секрет из события.

Спецификация функции

```
remove_secret(key: <строка>)
:: <пустое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
key	строка	Имя секрета для удаления.		да

Примеры

Удаление ключа API Datadog из события.

Пример 629. Исходный код

```
remove_secret("datadog_api_key")
```

Пример 630. Результат

```
null
```

set_secret

Устанавливает указанный секрет в событие.

Спецификация функции

```
set_secret(key: <строка>, secret: <строка>)
:: <пустое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`key`	строка	Имя секрета.		да
`secret`	строка	Значение секрета.		да

Примеры

Установка значения ключа API Datadog.

Пример 631. Исходный код

```
set_secret("datadog_api_key", "abc122")
```

Пример 632. Результат

```
null
```

set_semantic_meaning

Устанавливает семантический смысл для события. Обратите внимание, что эта функция задает значение при запуске Vector и *не имеет* поведения во время выполнения. Рекомендуется размещать все вызовы этой функции в начале функции VRL. Функцию нельзя вызывать с условием (например, нельзя предотвратить присваивание значения с использованием оператора if).

Спецификация функции

```
set_semantic_meaning(target: <путь>, meaning: <строка>)
:: <пустое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`target`	путь	Путь к значению, которому будет назначено семантическое значение.		да
`meaning`	строка	Имя назначаемого значения.		да

Примеры

Присваивание полю пользовательского семантического смысла

Пример 633. Исходный код

```
set_semantic_meaning(.foo, "bar")
```

Пример 634. Результат

```
null
```

Функции таблиц обогащения

find_enrichment_table_records

Поиск строк в таблице обогащения, соответствующих предоставленному условию.

Для таблиц обогащения типа `file` это условие должно быть объектом VRL, в котором ключи и значения указывают на поле для поиска и значение для поиска в этом поле. Эта функция возвращает строки, соответствующие предоставленным условиям. Все поля должны совпадать, чтобы строки были возвращены; если хотя бы одно поле не совпадает, строки не возвращаются.

В настоящее время существует две формы критериев поиска:

1. **Поиск точного совпадения**. Заданное поле должно точно совпадать со значением. Чувствительность к регистру может быть указана с помощью аргумента `case_sensitive`. Точный поиск может использовать индекс прямо в наборе данных, что делает его менее ресурсозатратным с точки зрения производительности.
2. **Поиск по диапазону дат**. Заданное поле должно быть больше или равно дате `from` и меньше или равно дате `to`. Обратите внимание, что поиск по диапазону дат включает последовательное сканирование строк, которые были найдены с помощью любых точных критериев. Такая операция может оказаться ресурсозатратной, если много строк возвращается любыми точными критериями. Рекомендуется использовать диапазоны дат как *единственные* критерии, когда набор данных обогащения очень мал.

Для таблиц обогащения `geoip` это условие должно быть объектом VRL с одной парой ключ-значение, значение которой должно быть действительным IP-адресом. Пример: `{"ip": .ip }`. Если ожидается

поле с результатом без значения, будет использовано `null`. Эта таблица может возвращать следующие поля:

```
find_enrichment_table_records(table: <строка>, condition: <объект>, [select: <массив>,
case_sensitive: <логическое значение>])
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`table`	строка	Таблица обогащения для поиска.		да
`condition`	объект	Условие для поиска. Поскольку условие используется при загрузке для создания индексов в данных, эти условия должны быть статически определены.		да
`select`	массив	Подмножество полей из таблицы обогащения, которые нужно вернуть. Если не указано, будут возвращены все поля.		нет
`case_sensitive`	логическое значение	Определяет, нужно ли точное совпадение текстовых полей.	`true`	нет

Примеры

Точное совпадение

Пример 635. Исходный код

```
find_enrichment_table_records!("test",
{"surname": "smith",},
case_sensitive: false)
```

Пример 636. Результат

```
[
  {
    "id": 1,
    "firstname": "Bob",
    "surname": "Smith"
  },
  {
    "id": 2,
    "firstname": "Fred",
    "surname": "Smith"
  }
]
```

Поиск по диапазону дат

Пример 637. Исходный код

```
find_enrichment_table_records!("test",
{
  "surname": "Smith",
  "date_of_birth": {
  "from": t'1985-01-01T00:00:00Z',
  "to": t'1985-12-31T00:00:00Z'
  }
})
```

Пример 638. Результат

```
[
  {
    "id": 1,
    "firstname": "Bob",
    "surname": "Smith"
  },
  {
    "id": 2,
    "firstname": "Fred",
    "surname": "Smith"
  }
]
```

get_enrichment_table_record

Функция ищет строку в таблице обогащения, которая соответствует указанному условию. Должна быть найдена одна строка. Если строка не найдена или найдено более одной строки, возвращается ошибка.

Для таблиц обогащения типа `file` это условие должно быть объектом VRL, в котором пары ключ-значение указывают поле для поиска, сопоставленное с искомым значением в этом поле. Эта функция возвращает строки, которые соответствуют предоставленным условиям. *Все* поля должны совпадать, чтобы строки были возвращены; если какие-либо поля не совпадают, строки не возвращаются.

На данный момент существует две формы критериев поиска:

1. **Поиск точного совпадения**. Указанное поле должно точно совпадать со значением. Регистрозависимость можно указать с помощью аргумента `case_sensitive`. Поиск точного совпадения может использовать индекс напрямую в наборе данных, что должно сделать этот поиск менее ресурсозатратным с точки зрения производительности.
2. **Поиск по диапазону дат**. Указанное поле должно быть больше или равно дате `from` и меньше или равно дате `to`. Обратите внимание, что поиск по диапазону даты включает последовательное сканирование строк, которые были найдены с помощью любых точных критериев. Такая операция может оказаться дорогостоящей, если много строк возвращается с помощью точных критериев. Рекомендуется использовать диапазоны дат как *единственные* критерии, когда объем

данных для обогащения очень мал.

Для таблиц обогащения `geoip` это условие должно быть объектом VRL с одной парой ключ-значение, значение которой должно быть допустимым IP-адресом. Пример: `{"ip": .ip }`. Если ожидается возвращаемое поле и оно не имеет значения, будет использовано значение `null`. Эта таблица может возвращать следующие поля:

- Базы данных ISP:
  - `autonomous_system_number`
  - `autonomous_system_organization`
  - `isp`
  - `organization`
- Базы данных City:
  - `city_name`
  - `continent_code`
  - `country_code`
  - `country_name`
  - `region_code`
  - `region_name`
  - `metro_code`
  - `latitude`
  - `longitude`
  - `postal_code`
  - `timezone`
- Базы данных Connection-Type:
  - `connection_type`

Спецификация функции

```
get_enrichment_table_record(table: <строка>, condition: <объект>, [select: <массив>,
case_sensitive: <логическое значение>])
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`table`	строка	Таблица обогащения для поиска.		да
`condition`	объект	Условие для поиска. Поскольку условие используется при запуске для создания индексов в данных, эти условия должны быть статически определены.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`select`	массив	Подмножество полей из таблицы обогащения для возврата. Если не указано, будут возвращены все поля.		нет
`case_sensitive`	логическое значение	Должны ли текстовые поля соответствовать регистру точно.	`true`	нет

Ошибки

Функция `get_enrichment_table_record` **возвращает ошибки**, для которых требуется обработка:

- Строка не найдена;
- Найдено несколько строк, соответствующих условию.

Эта ошибка не возвращается для искусственных таблиц обогащения в поле `setup_tests` элементов экспертизы.

Примеры

Точное совпадение

Пример 639. Исходный код

```
get_enrichment_table_record!("test",
{
  "surname": "bob",
  "firstname": "John"
},
case_sensitive: false)
```

Пример 640. Результат

```
{
  "id": 1,
  "firstname": "Bob",
  "surname": "Smith"
}
```

Поиск по диапазону даты

Пример 641. Исходный код

```
get_enrichment_table_record!("test",
{
  "surname": "Smith",
  "date_of_birth": {
    "from": t'1985-01-01T00:00:00Z',
    "to": t'1985-12-31T00:00:00Z'
```

```
  }
})
```

Пример 642. Результат

```
{
  "id": 1,
  "firstname": "Bob",
  "surname": "Smith"
}
```

Функции активных списков

add_active_record

Добавляет запись в активный список. Если запись с таким ключом уже существует, значение обновляется. В противном случае создается новая запись.

Спецификация функции

```
add_active_record(list: <строка>, value: <объект>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`list`	строка	название схемы активного списка, в который добавляется запись		да
`value`	объект	значение записи		да

Примеры

Добавление записи в активный список

Пример 643. Исходный код

```
add_active_record!("suspicious_ips", { "source_ip": .source_ip, "is_suspicious": true
})
```

Пример 644. Результат

```
true
```

Обновление записи в активном списке

Пример 645. Исходный код

```
failure_count = get_active_record!("auth_failure_counts", { "user_id": .user_id
```

```
}).failure_count || 0
add_active_record!("auth_failure_counts", { "user_id": .user_id, {"failure_count":
failure_count + 1 })
```

Пример 646. Результат

```
true
```

get_active_record

Извлекает значение из активного списка по указанному ключу. Возвращает значение, ассоциированное с ключом, или `null`, если запись отсутствует.

Спецификация функции

```
get_active_record(list: <строка>, key: <объект>)
:: <любое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`list`	строка	активный список, из которого нужно извлечь запись		да
`key`	объект	уникальный ключ записи		да

Примеры

Получение записи из активного списка

Пример 647. Исходный код

```
get_active_record!("allowed_ips", { "ip": .source_ip)
```

Пример 648. Результат

```
true
```

remove_active_record

Удаляет запись из активного списка. Если запись с таким ключом отсутствует, ничего не происходит.

Спецификация функции

```
remove_active_record(list: <строка>, key: <объект>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`list`	строка	активный список, из которого необходимо удалить запись		да
`key`	объект	уникальный ключ записи, которую необходимо удалить		да

Примеры

Удаление записи из активного списка

Пример 649. Исходный код

```
remove_active_record!("auth_failure_counts", { "user_id": .user_id })
```

Пример 650. Результат

```
true
```

Функции коллекций

В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции для работы с коллекциями предоставляют набор операций для манипуляции массивами и объектами.

Функции извлечения данных

includes

Определяет, включает ли массив `value` указанный элемент (`element`).

Спецификация функции

```
includes(value: <массив>, element: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив	Массив.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`element`	любое	Элемент для проверки.		да

Примеры

Массив включает

Пример 651. Исходный код

```
includes(["apple", "orange", "banana"], "banana")
```

Пример 652. Результат

```
true
```

keys

Возвращает ключи переданного в функцию объекта.

Спецификация функции

```
keys(value: <объект>)
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Объект, из которого нужно извлечь ключи.		да

Примеры

Получение ключей из объекта.

Пример 653. Исходный код

```
keys({"key1": "val1", "key2": "val2"})
```

Пример 654. Результат

```
[
  "key1",
  "key2"
]
```

length

Возвращает длину `value`. Результат зависит от типа `value`:

- для массива возвращает количество элементов;
- для объекта возвращает количество ключей на верхнем уровне;
- для строки возвращает количество байтов в строке. Если нужно получить количество символов, используйте функцию `strlen`.

Спецификация функции

```
length(value: <массив | объект | строка>)
:: <целое число>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив объект строка	Массив или объект		да

Примеры

Длина (объект)

Пример 655. Исходный код

```
length({
  "portland": "Trail Blazers",
  "seattle": "Supersonics"
})
```

Пример 656. Результат

```
2
```

Длина (вложенный объект)

Пример 657. Исходный код

```
length({
  "home": {
    "city": "Portland",
    "state": "Oregon"
  },
  "name": "Trail Blazers",
  "mascot": {
    "name": "Blaze the Trail Cat"
  }
```

```
})
```

Пример 658. Результат

```
3
```

Длина (массив)

Пример 659. Исходный код

```
length(["Trail Blazers", "Supersonics", "Grizzlies"])
```

Пример 660. Результат

```
3
```

Длина (строка)

Пример 661. Исходный код

```
length("The Planet of the Apes Musical")
```

Пример 662. Результат

```
30
```

values

Возвращает значения из переданного в функцию объекта.

Спецификация функции

```
values(value: <объект>)
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Объект, из которого извлекаются значения.		да

Примеры

Получить значения из объекта.

Пример 663. Исходный код

```
values({"key1": "val1", "key2": "val2"})
```

Пример 664. Результат

```
[
  "val1",
  "val2"
]
```

Чтобы получить срез массива, используйте функцию `slice`.

Функции фильтрации

compact

"Уплотняет" `value`, удаляя пустые элементы. В аргументах функции можно указать, какие значения следует считать пустыми.

Спецификация функции

```
compact(value: <массив | объект>, [recursive: <логическое значение>, null: <логическое
значение>, string: <логическое значение>, object: <логическое значение>, array:
<логическое значение>, nullish: <логическое значение>])
:: <массив | объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив объект	Объект или массив для уплотнения.		да
`recursive`	логическое значение	Будет ли уплотнение рекурсивным, то есть уплотняются ли элементы вложенных объектов и массивов.	`true`	нет
`null`	логическое значение	Должен ли `null` рассматриваться как пустое значение.	`true`	нет
`string`	логическое значение	Должна ли пустая строка рассматриваться как пустое значение.	`true`	нет
`object`	логическое значение	Должен ли пустой объект рассматриваться как пустое значение.	`true`	нет
`array`	логическое значение	Должен ли пустой массив рассматриваться как пустое значение.	`true`	нет

Аргумент	Тип	Описание	По умолчанию	Обязателен
`nullish`	логическое значение	Должны ли рассматриваться "нулевые" значения, которые определены функцией `is_nullish`, как пустые.	`false`	нет

Примеры

Сжатие массива

Пример 665. Исходный код

```
compact(["foo", "bar", "", null, [], "buzz"], string: true, array: true, null: true)
```

Пример 666. Результат

```
[
  "foo",
  "bar",
  "buzz"
]
```

Сжатие объекта

Пример 667. Исходный код

```
compact({"field1": 1, "field2": "", "field3": [], "field4": null}, string: true,
array: true, null: true)
```

Пример 668. Результат

```
{
  "field1": 1
}
```

filter

Отбирает элементы из коллекции.

Эта функция не поддерживает рекурсивную итерацию.

Функция использует синтаксис замыкания для доступа к ключу и значению или индексу и значению каждого элемента в коллекции.

К блокам замыкания применяются те же правила области видимости, что и к обычным блокам. Они могут использовать любые переменные, определенные в родительских областях видимости, а также сохранять изменения в этих переменных. Однако любые новые переменные, созданные внутри блока

замыкания, недоступны за его пределами.

Спецификация функции

```
filter(value: <массив | объект>) -> |<переменная ключа или индекса>, <переменная
значения>| {
    <блок>
}
:: <массив | объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив объект	Массив или объект для фильтрации.		да

Примеры

Фильтрация элементов

Пример 669. Исходный код

```
filter(array!(.tags)) -> |_index, value| {
# оставить элементы, которые не равны "foo"
value != "foo"
}
```

Пример 670. Результат

```
[
  "bar",
  "baz"
]
```

rv_coalesce_nullish

Возвращает первый элемент массива, значение которого не является "nullish", то есть элемент со значимым содержанием. "Nullish" считаются значения:

- `null`;
- `"-"`;
- состоящие только из пробелов, то есть символов со свойством Unicode `White_Space`.

Если все значения в массиве являются "nullish", функция возвращает `null`.

Спецификация функции

```
rv_coalesce_nullish(value: <массив>)
```

```
:: <любой тип>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив	массив, в котором производится поиск элементов, не являющихся "nullish"		да

Ошибки

Функция `rv_coalesce_nullish` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является массивом

Примеры

Есть и значения "nullish", и значимые

Пример 671. Исходный код

```
rv_coalesce_nullish(["  ", "-", "test"])
```

Пример 672. Результат

```
"test"
```

Все элементы имеют значимое содержание

Пример 673. Исходный код

```
rv_coalesce_nullish(["123132", "test"])
```

Пример 674. Результат

```
"123132"
```

Все элементы являются "nullish"

Пример 675. Исходный код

```
rv_coalesce_nullish(["-", "   ", "", null, ""])
```

Пример 676. Результат

```
null
```

unique

Возвращает уникальные значения для массива.

Сохраняется только первое вхождение каждого элемента.

Спецификация функции

```
unique(value: <массив>)
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив	Массив, из которого будут возвращены уникальные элементы.		да

Примеры

Уникальные значения

Пример 677. Исходный код

```
unique(["foo", "bar", "foo", "baz"])
```

Пример 678. Результат

```
[
  "foo",
  "bar",
  "baz"
]
```

Функции действий со всеми элементами

Чтобы избежать лишних вычислений, старайтесь выносить за пределы цикла все вычисления, не зависящие от него.

all

Проверяет, что все элементы `value` соответствуют условию, указанному в синтаксисе замыкания вида `|<переменная ключа>, <переменная значения>| { <блок> }`, где:

- в вертикальные черты заключены аргументы — ключ (для объектов) или индекс (для массивов) и значение элемента;
- в фигурных скобках указано собственно условие. В условии можно изменять внешние переменные.

Выполнение прекращается при первом значении `false` для любого элемента, дальнейшая проверка

не производится.

Спецификация функции

```
all(value: <массив или объект>) -> |<переменная ключа или индекс>, <переменная
значения>| { <блок> }
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив или объект	Массив или объект для проверки		да

Примеры

Итерация объекта

Пример 679. Исходный код

```
all({ "a": 1, "b": 2 }) -> |_key, value| { value > 0 }
```

Пример 680. Результат

```
true
```

Итерация массива

Пример 681. Исходный код

```
all([1, 2, 3, 4, 5]) -> |index, value| { value > 0 }
```

Пример 682. Результат

```
true
```

any

Проверяет, что хотя бы один элемент `value` соответствует условию, указанному в синтаксисе замыкания вида `|<переменная ключа>, <переменная значения>| { <блок> }`, где:

- в вертикальные черты заключены аргументы — ключ (для объектов) или индекс (для массивов) и значение элемента;
- в фигурных скобках указано собственно условие. В условии можно изменять внешние переменные.

Выполнение прекращается при первом значении `true` для любого элемента, дальнейшая проверка не производится.

Спецификация функции

```
any(value: <массив или объект>) -> |<переменная ключа или индекс>, <переменная
значения>| { <блок> }
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив или объект	Массив или объект для проверки		да

Примеры

Итерация объекта

Пример 683. Исходный код

```
any({ "a": 1, "b": 2 }) -> |_key, value| { value == 2 }
```

Пример 684. Результат

```
true
```

Итерация массива

Пример 685. Исходный код

```
all([1, 2, 3, 4, 5]) -> |index, value| { value == 2 }
```

Пример 686. Результат

```
true
```

for_each

Итерирует по коллекции.

В настоящее время функция **не поддерживает** рекурсивную итерацию.

Функция использует синтаксис замыкания для доступа к идентификатору и значению каждого элемента в коллекции.

К блокам замыкания применяются те же правила области видимости, что и к обычным блокам. Они могут использовать любые переменные, определенные в родительских областях видимости, а также сохранять изменения в этих переменных. Однако любые новые переменные, созданные внутри блока замыкания, недоступны за его пределами.

Спецификация функции

```
for_each(value: <массив | объект>) -> |<переменная ключа или индекса>, <переменная
значения>| {
    <блок>
}
:: <пустое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив объект	Массив или объект для итерации.		да

Примеры

Подсчет элементов

Пример 687. Исходный код

```
tally = {}
for_each(array!(.tags)) -> |_index, value| {
  # Получение текущего подсчета для `value` или установка его в `0`.
  count = int(get!(tally, [value])) ?? 0
  # Увеличение подсчета для значения на `1`.
  tally = set!(tally, [value], count + 1)
}
tally
```

Пример 688. Результат

```
{
  "foo": 2,
  "bar": 1,
  "baz": 1
}
```

map_keys

Отображает ключи объекта.

Если включен параметр `recursive`, функция проходит внутрь вложенных объектов согласно следующим правилам:

1. Итерация начинается с корневого объекта.
2. Для каждого вложенного объекта:
    - Сначала возвращается ключ самого объекта.
    - Затем происходит рекурсивный вызов функции для объекта, и происходит возврат к пункту 1.

- Любые изменения, сделанные во вложенном объекте **до** его рекурсивного вызова, сохраняются.

3. Для каждого вложенного массива:
   - Сначала возвращается ключ самого массива.
   - Затем происходит поиск всех объектов внутри массива и применение к каждому из них правил из пункта 2.

На практике эти правила означают, что `map_keys` с включенным параметром `recursive` найдет **все** ключи в целевом объекте, независимо от того, находятся ли вложенные объекты внутри массивов.

Функция использует синтаксис замыкания для доступа к ключу каждого элемента в объекте.

К блокам замыкания применяются те же правила области видимости, что и к обычным блокам. Они могут использовать любые переменные, определенные в родительских областях видимости, а также сохранять изменения в этих переменных. Однако любые новые переменные, созданные внутри блока замыкания, недоступны за его пределами.

Спецификация функции

```
map_keys(value: <объект>, [recursive: <логическое значение>]) -> |<переменная ключа>|
{ <блок> }
:: <объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Объект для итерации.		да
`recursive`	логическое значение	Определяет, следует ли выполнять рекурсивную итерацию по коллекции.	`false`	нет

Примеры

Преобразование ключей в верхний регистр

Пример 689. Исходный код

```
map_keys(.) -> |key| { upcase(key) }
```

Пример 690. Результат

```
{
  "FOO": "foo",
  "BAR": "bar"
}
```

Замена точек в ключах на подчеркивания

Пример 691. Исходный код

```
map_keys(., recursive: true) -> |key| { replace(key, ".", "_") }
```

Пример 692. Результат

```
{
  "labels": {
    "app_kubernetes_io/name": "mysql"
  }
}
```

map_values

Отображает значения внутри коллекции.

Если включен параметр `recursive`, функция проходит внутрь вложенных коллекций согласно следующим правилам:

1. Итерация начинается с корневой коллекции.
2. Для каждой вложенной коллекции:
    - Сначала возвращается сам тип коллекции.
    - Затем происходит рекурсивный вызов функции для коллекции, и происходит возврат к пункту 1 в этом списке.
    - Любые изменения, сделанные в коллекции **до** ее рекурсивного вызова, сохраняются.

Функция использует синтаксис замыкания, чтобы изменить значение каждого элемента в объекте.

К блокам замыкания применяются те же правила области видимости, что и к обычным блокам. Они могут использовать любые переменные, определенные в родительских областях видимости, а также сохранять изменения в этих переменных. Однако любые новые переменные, созданные внутри блока замыкания, недоступны за его пределами.

Спецификация функции

```
map_values(value: <массив | объект>, [recursive: <логическое значение>]) ->
|<переменная значения>| { <блок> }
:: <массив | объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив объект	Коллекция (массив или объект) для итерации.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`recursive`	логическое значение	Определяет, следует ли выполнять рекурсивную итерацию по коллекции.	`false`	нет

Примеры

Преобразование значений в верхний регистр

Пример 693. Исходный код

```
map_values(.) -> |value| { upcase!(value) }
```

Пример 694. Результат

```
{
  "foo": "FOO",
  "bar": "BAR"
}
```

match_datadog_query

Сопоставляет объект `value` с запросом Datadog Search Syntax.

Спецификация функции

```
match_datadog_query(value: <объект> , query: <строка> )
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Коллекция (объект)		да
`query`	строка	Запрос Datadog Search Syntax		да

Примеры

Запрос OR

Пример 695. Исходный код

```
match_datadog_query({"message": "contains this and that"}, "this OR that")
```

Пример 696. Результат

```
true
```

Запрос AND

Пример 697. Исходный код

```
match_datadog_query({"message": "contains only this"}, "this AND that")
```

Пример 698. Результат

```
false
```

Символ подстановки в атрибутах

Пример 699. Исходный код

```
match_datadog_query({"name": "foobar"}, "@name:foo*")
```

Пример 700. Результат

```
true
```

Диапазон тегов

Пример 701. Исходный код

```
match_datadog_query({"tags": ["a:x", "b:y", "c:z"]}, s'b:["x" TO "z"]')
```

Пример 702. Результат

```
true
```

Чтобы объединить все элементы массива в одну строку, используйте функцию `join`.

Функции изменения структуры

chunks

Разбивает `value` на части длиной `chunk_size` в байтах.

Спецификация функции

```
chunks(value: <массив | строка>, chunk_size: <целое число>)
:: <массив> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив строка	Массив байтов или строка для разделения.		да
`chunk_size`	целое число	Желаемая длина каждого фрагмента в байтах, не менее 1 байта. Длина может быть ограничена архитектурой хост-платформы.		да

Характеристики функции

При делении на фрагменты границы кодовых точек Unicode не соблюдаются.

Ошибки

Функция `chunks` **может возвращать ошибки**, для которых требуется обработка:

- `chunk_size` менее 1 байта.
- `chunk_size` слишком велик.

Примеры

Разделение строки на фрагменты

Пример 703. Исходный код

```
chunks("abcdefgh", 4)
```

Пример 704. Результат

```
[
  "abcd",
  "efgh"
]
```

Деление без учета границ кодовой точки Unicode

Пример 705. Исходный код

```
chunks("ab好", 4)
```

Пример 706. Результат

```
[
  "ab",
  "好"
]
```

flatten

Выполняет преобразование `value` в представление с одним уровнем.

Спецификация функции

```
flatten(value: <массив | объект>, [separator: <строка>])
:: <массив | объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив объект	Массив или объект, которые необходимо преобразовать.		да
`separator`	строка	Разделитель, используемый для объединения вложенных ключей.	`.`	нет

Примеры

Преобразование массива

Пример 707. Исходный код

```
flatten([1, [2, 3, 4], [5, [6, 7], 8], 9])
```

Пример 708. Результат

```
[
1,
2,
3,
4,
5,
6,
7,
8,
9
]
```

Преобразование объекта

Пример 709. Исходный код

```
flatten({
  "parent1": {
    "child1": 1,
    "child2": 2
  },
  "parent2": {
```

```
    "child3": 3
  }
})
```

Пример 710. Результат

```
{
  "parent1.child1": 1,
  "parent1.child2": 2,
  "parent2.child3": 3
}
```

unflatten

Преобразует `value` во вложенное представление. Функция может быть полезна для возвращения плоского объекта, полученного с помощью функции flatten, в исходную форму.

Спецификация функции

```
unflatten(value: <объект>, [separator: <строка>, recursive: <логическое значение>])
:: <объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Массив или объект, которые необходимо преобразовать.		да
`separator`	строка	Разделитель, используемый для разделения ключей.	`.`	нет
`recursive`	логическое значение	Требуется ли рекурсивно преобразовывать значения объекта.	`true`	нет

Примеры

Преобразование во вложенное представление

Пример 711. Исходный код

```
unflatten({
  "foo.bar.baz": true,
  "foo.bar.qux": false,
  "foo.quux": 42
})
```

Пример 712. Результат

```
{
  "foo": {
```

```
    "bar": {
      "baz": true,
      "qux": false
    },
    "quux": 42
  }
}
```

Рекурсивное преобразование во вложенное представление

Пример 713. Исходный код

```
unflatten({
  "flattened.parent": {
    "foo.bar": true,
    "foo.baz": false
  }
})
```

Пример 714. Результат

```
{
  "flattened": {
    "parent": {
      "foo": {
        "bar": true,
        "baz": false
      }
    }
  }
}
```

Нерекурсивное преобразование во вложенное представление

Пример 715. Исходный код

```
unflatten({
  "flattened.parent": {
    "foo.bar": true,
    "foo.baz": false
  }
},
recursive: false)
```

Пример 716. Результат

```
{
  "flattened": {
    "parent": {
```

```
      "foo.bar": true,
      "foo.baz": false
    }
  }
}
```

Игнорирование неконсистентных значений ключей

Если у нескольких значений один и тот же ключ является префиксом, значения ключей верхнего уровня вложенности отбрасываются, и сохраняется только вложенный объект.

В примере ниже для ключа `a` есть два возможных значения — число 3 и объект с ключами `b` и `c`. При применении функции `unflatten` числовое значение отбрасывается, сохраняется только объект.

Пример 717. Исходный код

```
unflatten({
  "a": 3,
  "a.b": 2,
  "a.c": 4
})
```

Пример 718. Результат

```
{
  "a": {
    "b": 2,
    "c": 4
  }
}
```

merge

Обновляет поля в объекте `to` на основе содержимого объекта `from`. При этом:

- Поля, которые есть только в одном из объектов, возвращаются в результирующем объекте без изменений.
- Если поле существует в обоих объектах, значение из объекта `from` заменяет значение из объекта `to`. Чтобы объединить при слиянии вложенные поля, используйте параметр `deep=true`.

Спецификация функции

```
merge(to: <объект>, from: <объект>, [deep: <логическое значение>])
:: <объект>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`to`	объект	Объект, в который выполняется слияние.		да
`from`	объект	Объект, из которого выполняется слияние.		да
`deep`	логическое значение	Выполняется глубокое слияние, если `true`, в противном случае слияние происходит только на верхнем уровне.	`false`	нет

Примеры

Поверхностное слияние объектов

Пример 719. Исходный код

```
merge(
  {
    "parent1": {
      "child1": 1,
      "child2": 2
    },
    "parent2": {
      "child3": 3
    }
  },
  {
  "parent1": {
    "child2": 4,
    "child5": 5
  }
})
```

Пример 720. Результат

```
{
  "parent1": {
    "child2": 4,
    "child5": 5
  },
  "parent2": {
    "child3": 3
  }
}
```

Глубокое слияние объектов

Пример 721. Исходный код

```
merge(
```

```
  {
    "parent1": {
      "child1": 1,
      "child2": 2
  },
   "parent2": {
     "child3": 3
   }
  },
  {
    "parent1": {
      "child2": 4,
      "child5": 5
    }
  },
deep: true
)
```

Пример 722. Результат

```
{
  "parent1": {
    "child1": 1,
    "child2": 4,
    "child5": 5
  },
   "parent2": {
    "child3": 3
  }
}
```

object_from_array

Создает объект из пар "ключ — значение", итерируя либо единый массив массивов, либо два отдельных массива. Ключи, равные `null`, игнорируются.

Если указан только один аргумент, он должен представлять собой массив, содержащий массивы с ключом и значением. Вместо пропущенных значений записывается `null`.

Если используются два массива, длина результирующего массива равна длине самого короткого входного массива, а все остальные элементы отбрасываются.

Спецификация функции

```
object_from_array(values: <массив>, [keys: <массив>])
:: <объект> , <error>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`values`	массив	Массив значений, или массив входных массивов, если других аргументов не передано.		да
`keys`	массив	Массив ключей. Если аргумент не указан, массивы и ключей, и значений должны быть переданы в первом аргументе.		нет

Ошибки

Функция `object_from_array` **может возвращать ошибки**, для которых требуется обработка:

- `values` или `keys` не являются массивами;
- `values` является единственным аргументом и содержит элементы, не являющиеся массивами.

Примеры

Создание объекта из массива массивов

Пример 723. Исходный код

```
object_from_array([["a", 1], [null, 2], ["b"], ["c", true, 3, 4]])
```

Пример 724. Результат

```
{
  "a": 1,
  "b": null,
  "c": true
}
```

Создание объекта из отдельных массивов ключей и значений

Пример 725. Исходный код

```
object_from_array([1, 2, 3, 4, 5], keys: ["one", null, "two"])
```

Пример 726. Результат

```
{
  "one": 1,
  "two": 3
}
```

rv_rename_keys

Переименовывает ключи в объекте согласно таблице обогащения.

Спецификация функции

```
rename_keys(value: <объект>, table: <строка>, from: <строка>, to <строка>, [separator:
<строка>, join_separator:  <строка>, case_sensitive: <логическое>])
:: <объект> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект	Объект, в котором нужно переименовать поля.		да
`table`	строка	Таблица обогащения.		да
`from`	строка	Столбец с текущими именами. Строка должна быть представлена литералом, без использования переменных.		да
`to`	строка	Столбец с новыми именами. Строка должна быть представлена литералом, без использования переменных.		да
`separator`	строка	Строка, разделяющая части сложных ключей. Строка должна быть представлена литералом, без использования переменных.		нет
`join_separator`	строка	Строка, соединяющая части сложных ключей после переименования. Если параметр не указан, то используется значение `separator`. Строка должна быть представлена литералом, без использования переменных.		нет
`case_sensitive`	логическое значение	Указывает, будет ли поиск ключей в таблице обогащения регистрозависимым. Значение должно быть представлено литералом (`true`/`false`), без использования переменных.	`false`	нет

Ошибки

Функция `rv_rename_keys` **может возвращать ошибки**, для которых требуется обработка:

- Не найдена таблица обогащения.
- Таблица не закодирована в UTF-8.
- `case_sensitive` не является логическим значением.
- `from` не является строкой
- `to` не является строкой
- `separator` не является строкой.
- `join_separator` не является строкой.
- Указан `join_separator` без параметра `separator`.
- Не удается разобрать таблицу.

Примеры

Пример 727. Исходный код

```
rename_keys!(
  { "a/b": 123 },
  table: "table_name",
  from: "from_value",
  to: "to_value",
  separator: "/"
)
```

Пример 728. Результат

```
{
  "c/d": 123
}
```

unnest

"Разворачивает" поле объекта, которое является массивом. В результате функция возвращает массив объектов, в каждом объекте значением указанном поля будет один из элементов массива, значения остальных полей сохраняются прежними. Это аналог "explode" в некоторых языках.

Присваивание возвращенного массива корневому пути `.` приведет к созданию нескольких событий из `remap`.

Спецификация функции

```
unnest(path: <путь>)
:: <массив> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`path`	путь	Путь к полю, которое нужно развернуть.		да

Ошибки

Функция `unnest` **может возвращать ошибки**, для которых требуется обработка:

- Поле, на которое ссылается путь, не является массивом.

Примеры

Развернуть поле

Пример 729. Исходный код

```
. = {"hostname": "localhost", "events": [{"message": "hello"}, {"message": "world"}]}
. = unnest(.events)
```

Пример 730. Результат

```
[
  {
    "events": {
      "message": "hello"
    },
    "hostname": "localhost"
  },
  {
    "events": {
      "message": "world"
    },
    "hostname": "localhost"
  }
]
```

Раскроить вложенное поле

Пример 731. Исходный код

```
. = unnest!(.event.messages)
```

zip

Объединяет элементы из нескольких итерируемых массивов в группы по одному элементу от каждого. Функция возвращает массив массивов, где каждый вложенный массив содержит элементы с одинаковыми индексами из всех входных массивов. Длина результирующего массива равна длине самого короткого входного массива, а все остальные элементы отбрасываются.

Функция аналогична функции `zip` в Python, а также методам `zip` в Ruby и Rust.

Если указан только один аргумент, он должен представлять собой массив, содержащий все массивы, которые нужно слить.

Спецификация функции

```
zip(array_0: <массив>, [array_1: <массив>])
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`array_0`	массив	Первый массив элементов, или массив входных массивов, если других аргументов не передано.		да
`array_1`	массив	Второй массив элементов. Если аргумент не указан, оба массива должны быть переданы в первом аргументе.		нет

Ошибки

Функция `zip` **может возвращать ошибки**, для которых требуется обработка:

- `array_0` или `array_1` не являются массивами;
- `array_0` является единственным аргументом и содержит элементы, не являющиеся массивами.

Примеры

Слияние двух массивов разной длины

Пример 732. Исходный код

```
zip([1, 2, 3], [4, 5, 6, 7])
```

Пример 733. Результат

```
[[1, 4], [2, 5], [3, 6]]
```

Слияние трех массивов

Пример 734. Исходный код

```
zip([[1, 2], [3, 4], [5, 6]])
```

Пример 735. Результат

```
[[1, 3, 5], [2, 4, 6]]
```

Функции действий с отдельными элементами

append

Добавляет каждый элемент из массива `items` в конец массива `value`.

Спецификация функции

```
append(value: <массив>, items: <массив>)
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
value	массив	Исходный массив		да
items	массив	Элементы для добавления		да

Примеры

Соединение массивов

Пример 736. Исходный код

```
append([1, 2], [3, 4])
```

Пример 737. Результат

```
[
  1,
  2,
  3,
  4
]
```

del

Удаляет из цели поле, указанное статическим path.

Для удаления динамического пути см. функцию remove.

Спецификация функции

```
del(path: <путь>, [compact: <логическое значение>])
:: <любой тип>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
path	путь	Путь к удаляемому полю.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`compact`	логическое значение	Если `compact` установлен в true, после удаления, если остался пустой объект или массив, он также должен быть удален, рекурсивно вплоть до корневого элемента. Это применяется только к удаляемому пути и любым родительским путям.	`false`	нет

Примечания

У этой функции есть особое поведение, о котором стоит знать:

- Функция `del` модифицирует текущее событие и возвращает значение удаленного поля.

Примеры

Удаление поля

Пример 738. Исходный код

```
del(.field1)
```

Переименование поля

Пример 739. Исходный код

```
.new_field = del(.old_field)
```

exists

Проверяет, существует ли путь в цели.

Эта функция различает отсутствующий путь и путь со значением `null`. Обычное выражение пути, такое как `.foo`, не может различить эти случаи, так как он всегда возвращает значение `null`, если путь не существует.

Спецификация функции

```
exists(path: <путь>)
:: <логическое значение>
```

Примеры

Проверка существования поля

Пример 740. Исходный код

```
exists(.field)
```

Пример 741. Результат

```
true
```

Проверка существования элемента массива

Пример 742. Исходный код

```
exists(.array[2])
```

Пример 743. Результат

```
true
```

get

Динамически получает значение по указанному пути.

Когда вы знаете путь, который вы хотите найти, вы должны использовать статические пути, такие как `.foo.bar[1]`, чтобы получить значение этого пути. Однако, когда вы не знаете заранее имена путей, вы можете использовать динамическую функцию `get`, чтобы получить значение по запросу.

Спецификация функции

```
get(value: <объект | массив>, path: <массив>)
:: <любой тип>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект массив	Объект или массив для запроса.		да
`path`	массив	Массив из сегментов пути для поиска значения.		да

Ошибки

Функция `get` **может возвращать ошибки**, для которых требуется обработка:

- Сегмент пути должен быть либо строкой, либо целым числом.

Примеры

Поле верхнего уровня с одним сегментом

Пример 744. Исходный код

```
get!(value: { "foo": "bar" }, path: ["foo"])
```

Пример 745. Результат

```
bar
```

Вложенное поле с несколькими сегментами

Пример 746. Исходный код

```
get!(value: { "foo": { "bar": "baz" } }, path: ["foo", "bar"])
```

Пример 747. Результат

```
baz
```

Индексирование массива

Пример 748. Исходный код

```
get!(value: ["foo", "bar", "baz"], path: [-2])
```

Пример 749. Результат

```
bar
```

push

Добавляет `item` в конец массива `value`.

Спецификация функции

```
push(value: <массив>, item: <любой тип>)
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив	Целевой массив.		да
`item`	любое	Элемент для добавления.		да

Примеры

Добавление элемента в массив

Пример 750. Исходный код

```
push([1, 2], 3)
```

Пример 751. Результат

```
[
  1,
  2,
  3
]
```

Добавление массива в массив

Пример 752. Исходный код

```
push([1, 2], [3, 4])
```

Пример 753. Результат

```
[
  1,
  2,
  [
    3,
    4
  ]
]
```

remove

Динамически удаляет значение по указанному пути.

Когда вы знаете путь, который хотите удалить, используйте функцию `del` и статические пути, такие как `del(.foo.bar[1])`, чтобы удалить значение по этому пути. Функция `del` возвращает удаленное значение и является более производительной, чем эта функция. Однако, когда вы не знаете заранее имена путей, вы можете использовать эту динамическую функцию для удаления значения по указанному пути.

Спецификация функции

```
remove(value: <объект | массив>, path: <массив>, [compact: <логическое значение>])
:: <объект | массив>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект массив	Объект или массив, из которого требуется удалить данные.		да
`path`	массив	Массив сегментов пути для удаления значения.		да
`compact`	логическое значение	Определяет, следует ли после удаления удалять пустые объекты или массивы.	`false`	нет

Ошибки

Функция `remove` **может возвращать ошибки**, для которых требуется обработка:

- сегмент пути должен быть либо "строкой", либо "целым числом"

Примеры

Удаление значения односегментного поля верхнего уровня

Пример 754. Исходный код

```
remove!(value: { "foo": "bar" }, path: ["foo"])
```

Пример 755. Результат

```
{}
```

Удаление значения многосегментного вложенного поля

Пример 756. Исходный код

```
remove!(value: { "foo": { "bar": "baz" } }, path: ["foo", "bar"])
```

Пример 757. Результат

```
{
  "foo": {}
}
```

Индексирование массива

Пример 758. Исходный код

```
remove!(value: ["foo", "bar", "baz"], path: [-2])
```

Пример 759. Результат

```
[
  "foo",
  "baz"
]
```

Удаление с уплотнением

Пример 760. Исходный код

```
remove!(value: { "foo": { "bar": [42], "baz": true } }, path: ["foo", "bar", 0],
compact: true)
```

Пример 761. Результат

```
{
  "foo": {
    "baz": true
  }
}
```

set

Динамически вставляет данные в указанный путь объекта или массива.

Когда вы знаете путь, которому хотите присвоить значение, следует использовать статические пути, такие как `.foo.bar[1] = true`, для улучшения производительности и читаемости. Однако, когда вы не знаете заранее имена путей, вы можете использовать эту динамическую функцию вставки для вставки данных в объект или массив.

Спецификация функции

```
set(value: <объект | массив>, path: <массив>, data: <любой тип>)
:: <объект | массив>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект массив	Объект или массив для вставки данных.		да
`path`	массив	Массив сегментов пути для вставки значения.		да
`data`	любое	Вставляемые данные.		да

Ошибки

Функция `set` **может возвращать ошибки**, для которых требуется обработка:

- сегмент пути должен быть либо "строкой", либо "целым числом"

Примеры

Вставка значения в односегментное поле верхнего уровня

Пример 762. Исходный код

```
set!(value: { "foo": "bar" }, path: ["foo"], data: "baz")
```

Пример 763. Результат

```
{
  "foo": "baz"
}
```

Вставка значения в многосегментное вложенное поле

Пример 764. Исходный код

```
set!(value: { "foo": { "bar": "baz" } }, path: ["foo", "bar"], data: "qux")
```

Пример 765. Результат

```
{
  "foo": {
    "bar": "qux"
  }
}
```

Массив

Пример 766. Исходный код

```
set!(value: ["foo", "bar", "baz"], path: [-2], data: 42)
```

Пример 767. Результат

```
[
  "foo",
  42,
  "baz"
]
```

Функции строк

В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции поиска и сопоставления

Выбирайте функцию в зависимости от количества условий и требований к производительности.

contains

Определяет, содержит ли строка `value` указанную подстроку `substring`.

Чтобы проверить наличие нескольких подстрок, используйте функцию `rv_contains_any`. Функция вернет `true`, если хотя бы одна из переданных подстрок присутствует в строке.

Спецификация функции

```
contains(value: <строка>, substring: <строка>, [case_sensitive: <логическое
значение>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Текст для поиска.		да
`substring`	строка	Подстрока, которую нужно найти в `value`.		да
`case_sensitive`	логическое значение	Должен ли поиск учитывать регистр символов.	`true`	нет

Примеры

Проверка наличия подстроки в строке (с учетом регистра)

Пример 768. Исходный код

```
contains("The Needle In The Haystack", "Needle")
```

Пример 769. Результат

```
true
```

Проверка наличия подстроки в строке (без учета регистра)

Пример 770. Исходный код

```
contains("The Needle In The Haystack", "needle", case_sensitive: false)
```

Пример 771. Результат

```
true
```

contains_all

Определяет, содержит ли строка `value` все указанные подстроки (`substrings`).

Чтобы проверить наличие в строке хотя бы одной из подстрок, используйте функцию `rv_contains_any`. Функция вернет `true`, если одна из переданных подстрок присутствует в строке.

Спецификация функции

```
contains_all(value: <строка>, substrings: <массив>, [case_sensitive: <логическое
значение>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Текст для поиска.		да
`substrings`	массив	Массив подстрок, которые нужно найти в `value`.		да
`case_sensitive`	логическое значение	Должен ли поиск учитывать регистр символов.	`true`	нет

Примеры

Проверка наличия подстрок в строке (с учетом регистра)

Пример 772. Исходный код

```
contains_all("The Needle In The Haystack", ["Needle", "Haystack"])
```

Пример 773. Результат

```
true
```

Проверка наличия подстроки в строке (без учета регистра)

Пример 774. Исходный код

```
contains_all("the NEEDLE in the haystack", ["needle", "haystack"], case_sensitive:
false)
```

Пример 775. Результат

```
true
```

rv_contains_any

Определяет, содержится ли в строке `value` хотя бы одна подстрока из массива `substrings`.

Массив строк должен быть представлен литералом, без использования переменных. Это позволяет оптимизировать выполнение, заранее построив структуру для поиска конкретных подстрок.

Спецификация функции

```
rv_contains_any(value: <строка>, substrings: <массив>, [case_sensitive: <логическое
значение>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, в которой производится поиск подстрок.		да
`substrings`	массив	Литерал массива строк, которые ищутся в `value`.		да
`case_sensitive`	логическое значение	Определяет, учитывается ли регистр символов при поиске.	`true`	нет

Ошибки

Функция `rv_contains_any` **может возвращать ошибки**, для которых требуется обработка:

- Подстроки содержат недопустимые данные.

Примеры

Одна из подстрок содержится в значении

Пример 776. Исходный код

```
rv_contains_any("eqwas_)xz!123", ["asd", "xz"])
```

Пример 777. Результат

```
true
```

Одна из подстрок содержится в значении без учета регистра

Пример 778. Исходный код

```
rv_contains_any("eqwas_)xZ!123", ["asd", "Xz"], case_sensitive: false)
```

Пример 779. Результат

```
true
```

Ни одна подстрока не содержится в значении

Пример 780. Исходный код

```
rv_contains_any("eqwas_)xz!123", ["asd", "xz1"])
```

Пример 781. Результат

```
false
```

starts_with

Определяет, начинается ли `value` с подстроки `substring`.

Чтобы проверить несколько подстрок, используйте функцию `rv_starts_with_any`. Функция вернет `true`, если строка начинается хотя бы с одной из переданных подстрок.

Спецификация функции

```
starts_with(value: <строка>, substring: <строка>, [case_sensitive: <логическое
значение>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для поиска.		да
`substring`	строка	Подстрока, с которой `value` должно начинаться.		да
`case_sensitive`	логическое значение	Определяет, должен ли поиск быть чувствительным к регистру.	`true`	нет

Примеры

Строка начинается с подстроки (с учетом регистра)

Пример 782. Исходный код

```
starts_with("The Needle In The Haystack", "The Needle")
```

Пример 783. Результат

```
true
```

Строка начинается с подстроки (без учета регистра)

Пример 784. Исходный код

```
starts_with("The Needle In The Haystack", "the needle", case_sensitive: false)
```

Пример 785. Результат

```
true
```

rv_starts_with_any

Определяет, начинается ли строка `value` хотя бы с одной подстроки из массива `substrings`.

Массив строк должен быть представлен литералом, без использования переменных. Это позволяет оптимизировать выполнение, заранее построив структуру для поиска конкретных подстрок.

Спецификация функции

```
rv_starts_with_any(value: <строка>, substrings: <массив>, [case_sensitive: <логическое значение>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, в которой производится поиск подстрок.		да
`substrings`	массив	Литерал массива строк, которые ищутся в `value`.		да
`case_sensitive`	логическое значение	Определяет, учитывается ли регистр символов при поиске.	`true`	нет

Ошибки

Функция `rv_contains_any` **может возвращать ошибки**, для которых требуется обработка:

- Подстроки содержат недопустимые данные.

Примеры

Значение начинается с одной из подстрок

Пример 786. Исходный код

```
rv_starts_with_any("xzeqwas_)xz!123", ["asd", "xz"])
```

Пример 787. Результат

```
true
```

Значение начинается с одной из подстрок без учета регистра

Пример 788. Исходный код

```
rv_starts_with_any("xZeqwas_)xz!123", ["asd", "Xz"], case_sensitive: false)
```

Пример 789. Результат

```
true
```

Значение не начинается ни с одной из подстрок

Пример 790. Исходный код

```
rv_starts_with_any("xzeqwas_)xz1!123", ["asd", "xz1"])
```

Пример 791. Результат

```
false
```

ends_with

Определяет, заканчивается ли строка `value` указанной подстрокой `substring`.

Чтобы проверить несколько подстрок, используйте функцию `rv_ends_with_any`. Функция вернет `true`, если строка заканчивается хотя бы одной из переданных подстрок.

Спецификация функции

```
ends_with(value: <строка>, substring: <строка>, [case_sensitive: <логическое
значение>])
```

```
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для поиска.		да
`substring`	строка	Подстрока, которой должно заканчиваться `value`.		да
`case_sensitive`	логическое значение	Определяет, должен ли поиск быть чувствительным к регистру.	`true`	нет

Примеры

Строка заканчивается (с учетом регистра)

Пример 792. Исходный код

```
ends_with("The Needle In The Haystack", "The Haystack")
```

Пример 793. Результат

```
true
```

Строка заканчивается (без учета регистра)

Пример 794. Исходный код

```
ends_with("The Needle In The Haystack", "the haystack", case_sensitive: false)
```

Пример 795. Результат

```
true
```

rv_ends_with_any

Определяет, заканчивается ли строка `value` хотя бы одной подстрокой из массива `substrings`.

Массив строк должен быть представлен литералом, без использования переменных. Это позволяет оптимизировать выполнение, заранее построив структуру для поиска конкретных подстрок.

Спецификация функции

```
rv_ends_with_any(value: <строка>, substrings: <массив>, [case_sensitive: <логическое
значение>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, в которой производится поиск подстрок.		да
`substrings`	массив	Литерал массива строк, которые ищутся в `value`.		да
`case_sensitive`	логическое значение	Определяет, учитывается ли регистр символов при поиске.	`true`	нет

Ошибки

Функция `rv_contains_any` **может возвращать ошибки**, для которых требуется обработка:

- Подстроки содержат недопустимые данные.

Примеры

Значение заканчивается одной из подстрок

Пример 796. Исходный код

```
rv_ends_with_any("eqwas_)xz!123xz", ["asd", "xz"])
```

Пример 797. Результат

```
true
```

Значение заканчивается одной из подстрок без учета регистра

Пример 798. Исходный код

```
rv_ends_with_any("eqwas_)xz!123xZ", ["asd", "Xz"], case_sensitive: false)
```

Пример 799. Результат

```
true
```

Значение не заканчивается ни одной из подстрок

Пример 800. Исходный код

```
rv_ends_with_any("eqwas_)xz!123xz", ["asd", "xz1"])
```

Пример 801. Результат

```
false
```

find

Определяет позицию начала первого найденного элемента в строке `value`, слева направо, который соответствует шаблону `pattern`, или возвращает -1, если элемент не найден.

Спецификация функции

```
find(value: <строка>, pattern: <регулярное выражение | строка>, [from: <целое число>])
:: <целое число>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для поиска паттерна.		да
`pattern`	регулярное выражение строка	Регулярное выражение или строка для сравнения.		да
`from`	целое число	Смещение, с которого начинается поиск.	`0`	нет

Примеры

Поиск текста

Пример 802. Исходный код

```
find("foobar", "foo")
```

Пример 803. Результат

```
0
```

Поиск по регулярному выражению

Пример 804. Исходный код

```
find("foobar", r'b.r')
```

Пример 805. Результат

```
3
```

Нет совпадений

Пример 806. Исходный код

```
find("foobar", "baz")
```

Пример 807. Результат

```
-1
```

С учетом смещения

Пример 808. Исходный код

```
find("foobarfoobarfoo", "bar", 4)
```

Пример 809. Результат

```
9
```

match

Определяет, соответствует ли `value` указанному шаблону `pattern`.

Спецификация функции

```
match(value: <строка>, pattern: <регулярное выражение>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Значение для проверки на соответствие.		да
`pattern`	регулярное выражение	Регулярное выражение для проверки соответствия.		да

Примеры

Проверка на соответствие регулярному выражению

Пример 810. Исходный код

```
match("I'm a little teapot", r'teapot')
```

Пример 811. Результат

```
true
```

match_any

Определяет, соответствует ли `value` хотя бы одному из указанных шаблонов `patterns`. Все шаблоны проверяются за один проход по целевой строке, что дает этой функции преимущество в производительности перед многократным вызовом функции `match`.

Спецификация функции

```
match_any(value: <строка>, patterns: <массив>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Значение для проверки на соответствие.		да
`patterns`	массив	Массив регулярных выражений для проверки соответствия.		да

Примеры

Проверка на соответствие хотя бы одному регулярному выражению

Пример 812. Исходный код

```
match_any("I'm a little teapot", [r'frying pan', r'teapot'])
```

Пример 813. Результат

```
true
```

Функции анализа строк

shannon_entropy

Вычисляет энтропию Шеннона для заданной строки. Функция может вычислять энтропию на основе байтов строки, кодовых точек или графем.

Разбиение на байты — самое быстрое, но может дать нежелательные результаты при работе с UTF-8. Разбиение на графемы — самое медленное, но наиболее точное в таких случаях.

Спецификация функции

```
shannon_entropy(value: <строка>, [segmentation: <string>])
:: <число с плавающей точкой>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Входная строка.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`segmentation`	строка	Способ разбиения строки для вычисления энтропии. Допустимые значения: • `"bytes"`; • `"codepoints"`; • `"graphemes"`.	`"bytes"`	нет

Примеры

Вычисление энтропии с разбиением строки на байты

Пример 814. Исходный код

```
floor(shannon_entropy("vector.dev"), precision: 4)
```

Пример 815. Результат

```
2.9219
```

Вычисление энтропии строки в UTF-8 с разбиением на байты

Пример 816. Исходный код

```
floor(shannon_entropy("test123%456.□□□□□.net."), precision: 4)
```

Пример 817. Результат

```
4.0784
```

Вычисление энтропии строки в UTF-8 с разбиением на графемы

Пример 818. Исходный код

```
floor(shannon_entropy("test123%456.□□□□□.net.", segmentation: "grapheme"), precision: 4)
```

Пример 819. Результат

```
3.9362
```

strlen

Возвращает количество символов UTF-8 в строке `value`. Это отличается от `length`, который бы подсчитал количество байтов в строке.

Обратите внимание, что это подсчет значений Unicode, который иногда может отличаться от

кодовых точек Unicode.

Спецификация функции

```
strlen(value: <строка>)
:: <целое число>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка		да

Примеры

strlen

Пример 820. Исходный код

```
strlen("ñandú")
```

Пример 821. Результат

```
5
```

Функции разбиения и слияния массивов и строк

join

Объединяет каждую строку в массиве `value` в одну строку с возможным разделителем `separator`.

Чтобы упростить структуру кода, вместо функции `join` можно использовать шаблонизацию строк.

Спецификация функции

```
join(value: <массив>, [separator: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив	Массив строк, которые нужно объединить.		да
`separator`	строка	Строка, разделяющая каждый элемент при объединении.		нет

Примеры

Объединение массива (без разделителя)

Пример 822. Исходный код

```
join!(["bring", "us", "together"])
```

Пример 823. Результат

```
bringustogether
```

Объединение массива (разделитель — запятая)

Пример 824. Исходный код

```
join!(["sources", "transforms", "sinks"], separator: ", ")
```

Пример 825. Результат

```
sources, transforms, sinks
```

slice

Возвращает срез массива или строки `value` между позициями `start` и `end`.

Если параметры `start` и `end` отрицательные, они отсчитываются справа от строки или массива. Если `end` указывает на позицию, которая больше длины строки или массива, возвращается срез до конца строки или массива.

Спецификация функции

```
slice(value: <массив | строка>, start: <целое число>, [end: <целое число>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	массив строка	Строка или массив для среза.		да
`start`	целое число	Включительная начальная позиция. Индекс с нуля, который может быть отрицательным.		да
`end`	целое число	Исключительная конечная позиция. Индекс с нуля, который может быть отрицательным.	длина строки	нет

Примеры

Срез строки (положительный индекс)

Пример 826. Исходный код

```
slice!("Supercalifragilisticexpialidocious", start: 5, end: 13)
```

Пример 827. Результат

```
califrag
```

Срез строки (отрицательный индекс)

Пример 828. Исходный код

```
slice!("Supercalifragilisticexpialidocious", start: 5, end: -14)
```

Пример 829. Результат

```
califragilistic
```

split

Разделяет строку `value` с использованием шаблона `pattern`.

Спецификация функции

```
split(value: <строка>, pattern: <строка | регулярное выражение>, [limit: <целое
число>])
:: <массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разделения.		да
`pattern`	строка регулярное выражение	Строка разделяется каждый раз, когда совпадает с этим шаблоном. Может быть строкой или регулярным выражением.		да
`limit`	целое число	Максимальное количество подстрок для возврата.		нет

Примеры

Разделение строки (без ограничения)

Пример 830. Исходный код

```
split("apples and pears and bananas", " and ")
```

Пример 831. Результат

```
[
  "apples",
  "pears",
  "bananas"]
```

Разделение строки (с ограничением)

Пример 832. Исходный код

```
split("apples and pears and bananas", " and ", limit: 2)
```

Пример 833. Результат

```
[
  "apples",
  "pears and bananas"
]
```

Чтобы разбить строку на части заданной длины в байтах, используйте функцию `chunks`.

Функции редактирования строк

redact

Скрывает конфиденциальные данные, такие как номера социального страхования США (SSN) и другие виды личной информации, с помощью пользовательских шаблонов. Это помогает соблюдать правила и гарантировать, что конфиденциальные данные не покидают вашу сеть.

Поведение функции зависит от типа значения `value`:

- Для строк — скрывает конфиденциальные данные и возвращает новую строку.
- Для массивов — скрывает конфиденциальные данные в каждом элементе строки.
- Для объектов — маскирует конфиденциальные данные в каждом значении строки, но не в ключах.

В массивах и объектах функция рекурсивно применяет эти операции ко всем вложенным массивам и объектам. Любые нестроковые элементы будут пропущены.

Любой скрытый текст будет заменен на `[REDACTED]`.

Спецификация функции

```
redact(value: <строка | объект | массив>, filters: <массив>, [redactor: <строка |
объект> ]))
```

```
:: <строка | объект | массив>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка объект массив	Значение, в котором требуется скрыть конфиденциальные данные.		да
`filters`	массив	Список фильтров, которые будут применены к `value`. Каждый фильтр может быть указан одним из следующих способов: • В виде регулярного выражения, которое будет использовано для скрытия соответствующего ему текста. • В виде объекта `{"type": "pattern", "patterns": <массив>}`, где под ключом `patterns` указан массив регулярных выражений. Именованный фильтр `us_social_security_number` скрывает номера социального страхования в США. Он может быть двумя способами: • в виде объекта `{"type": "us_social_security_number"}`; • в виде строки `"us_social_security_number"`. Данный параметр должен быть представлен литералом, без использования переменных или других динамических выражений. Это позволяет выполнить проверку аргумента на этапе компиляции и исключить ошибки во время выполнения.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`redactor`	строка объект	Указывает, на что заменить совпадения с фильтрами. Аргумент задается как объект вида `{"type": <строка>, [<дополнительные ключи>]}`, где указаны тип используемой замены под ключом `type` и дополнительные ключи в зависимости от типа. Поддерживаются следующие типы: • `full` — замена совпадений на строку `"[REDACTED]"`. • `text` — замена на произвольную строку, указанную под обязательным ключом `replacement`. • `sha2` — хеширование совпадений по алгоритму SHA-2. Поддерживает два необязательных параметра: ◦ `variant` — вариант используемого алгоритма: `SHA-224`, `SHA-256`, `SHA-384`, `SHA-512`, `SHA-512/224`, `SHA-512/256` (по умолчанию); ◦ `encoding` — кодировка хеша в виде текста: `base16` или `base64` (по умолчанию). • `sha3` — хеширование совпадений по алгоритму SHA-3. Поддерживает два необязательных параметра: ◦ `variant` — вариант используемого алгоритма: `SHA-224`, `SHA-256`, `SHA-384`, `SHA-512` (по умолчанию); ◦ `encoding` — кодировка хеша в виде текста: `base16` или `base64` (по умолчанию). Этот параметр должен быть представлен статическим выражением. Нельзя использовать переменные или другие динамические выражения. Это позволяет выполнить проверку аргумента на этапе компиляции и исключить ошибки во время выполнения.	`"full"`	нет

Для часто используемых шаблонов замены можно использовать сокращения в

виде строк:

- `"full"` эквивалентна записи `{"type": "full"}`
- `"sha2"` эквивалентна записи `{"type": "sha2", "variant": "SHA-512/256", "encoding": "base64"}`
- `"sha3"` эквивалентна записи `{"type": "sha3", "variant": "SHA3-512", "encoding": "base64"}`

Примеры

Замена текста с помощью регулярного выражения

Пример 834. Исходный код

```
redact("my id is 123456", filters: [r'\d+'])
```

Пример 835. Результат

```
my id is [REDACTED]
```

Замена номеров социального страхования США в любом поле

Пример 836. Исходный код

```
redact({ "name": "John Doe",
"ssn": "123-12-1234"},
filters: ["us_social_security_number"])
```

Пример 837. Результат

```
{
  "name": "John Doe",
  "ssn": "[REDACTED]"
}
```

Замена произвольным текстом

Пример 838. Исходный код

```
redact("my id is 123456",
filters: [r'\d+'],
redactor: {"type": "text", "replacement": "***"})
```

Пример 839. Результат

```
my id is ***
```

Замена хешем SHA-3

Пример 840. Исходный код

```
redact("my id is 123456",
filters: [r'\d+'],
redactor: "sha3")
```

Пример 841. Результат

```
my id is
ZNCdmTDI7PeeUTFnpYjLdUObdizo+bIupZdl8yqnTKGdLx6X3JIqPUlUWUoFBikX+yTR+OcvLtAqWO11NPlNJw
==
```

Замена хешем SHA-256 в шестнадцатеричной кодировке

Пример 842. Исходный код

```
redact("my id is 123456",
filters: [r'\d+'],
redactor: {"type": "sha2",
"variant": "SHA-256",
"encoding": "base16"})
```

Пример 843. Результат

```
my id is 8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92
```

replace

Заменяет все совпадения с шаблоном `pattern` из значения `value` на строку `with`.

В аргументе `with` могут использоваться именованные группы.

Спецификация функции

```
replace(value: <строка>, pattern: <регулярное выражение | строка>, with: <строка>,
[count: <целое число>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Исходная строка.		да
`pattern`	регулярное выражение строка	Заменяет все совпадения с этим шаблоном. Может быть статической строкой или регулярным выражением.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`with`	строка	Строка, которой заменяются совпадения.		да
`count`	целое число	Максимальное количество замен. `-1` означает замену всех совпадений.	`-1`	нет

Примеры

Замена текста

Пример 844. Исходный код

```
replace("Apples and Bananas", "and", "not")
```

Пример 845. Результат

```
Apples not Bananas
```

Замена с помощью регулярного выражения

Пример 846. Исходный код

```
replace("Apples and Bananas", r'(?i)bananas', "Pineapples")
```

Пример 847. Результат

```
Apples and Pineapples
```

Замена только первого совпадения

Пример 848. Исходный код

```
replace("Bananas and Bananas", "Bananas", "Pineapples", count: 1)
```

Пример 849. Результат

```
Pineapples and Bananas
```

Замена с захватом групп

Пример 850. Исходный код

```
replace("foo123bar", r'foo(?P<num>\d+)bar', "$num")
```

Пример 851. Результат

```
"123"
```

replace_with

Заменяет все совпадения с шаблоном `pattern` из строки `value` с помощью замыкания.

Аргумент `pattern` принимает регулярное выражение, которое может использовать захват групп.

Синтаксис замыкания используется, чтобы вычислить строку, на которую заменяются совпадения. Замыкание принимает единственный аргумент, представляющий собой объект со следующими полями:

- `captures` — массив, элементами которого являются захваченные группы.

  Если группа является незахватывающей (то есть обозначена `?:`) то значение соответствующего элемента равно `null`.

- `string` — строка, включающая полное совпадение.
- поля, соответствующие именованным группам.

В результате функции возвращается новая строка с заменой совпадений, исходная строка не изменяется.

Спецификация функции

```
replace_with(value: <строка>, pattern: <регулярное выражение>, [count: <целое число>
]) -> |<переменная>| { <блок> }
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Исходная строка.		да
`pattern`	регулярное выражение	Шаблон для поиска совпадений. Может содержать захват групп. Нельзя использовать имена групп `captures` и `string`.		да
`count`	целое число	Максимальное количество замен. `-1` означает замену всех совпадений.	`-1`	нет

Примеры

Сделать первые буквы слов прописными

Пример 852. Исходный код

```
replace_with("no server available",
r'\b(\w)(\w*)') -> |match| {
    upcase!(match.captures[0]) + string!(match.captures[1])
}
```

Пример 853. Результат

```
No Server Available
```

Замена на хэш

Пример 854. Исходный код

```
replace_with("email from test@example.com", r'\w+@example.com') -> |match| {
    sha2(match.string, variant: "SHA-512/224")
}
```

Пример 855. Результат

```
email from adf6e1bc4415d24912bd93072ad34ef825a7b6eb3bf53f68def1fc17
```

Замена на хэш

Пример 856. Исходный код

```
replace_with("email from test@example.com", r'\w+@example.com') -> |match| {
    sha2(match.string, variant: "SHA-512/224")
}
```

Пример 857. Результат

```
email from adf6e1bc4415d24912bd93072ad34ef825a7b6eb3bf53f68def1fc17
```

Замена первого вхождения

Пример 858. Исходный код

```
replace_with("The file was not found, check the file name.", r'(?i)file', count: 1) ->
|match| {
    "object"
}
```

Пример 859. Результат

```
The object was not found, check the file name.
```

Именованная группа

Пример 860. Исходный код

```
replace_with("level=error A message", r'level=(?P<level>\w+)') -> |match| {
    lvl = upcase!(match.level)
    "[{{lvl}}]"
}
```

Пример 861. Результат

```
[ERROR] A message
```

sieve

Сохраняет в `value` только совпадения с шаблоном `pattern`.

Функцию можно использовать для определения символов, которые разрешены в строке, и удаления всего остального.

Спецификация функции

```
sieve(value: <строка>, pattern: <регулярное выражение>, [replace_single: <строка> ,
replace_repeated: <строка> ])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Исходная строка.		да
`pattern`	регулярное выражение	Шаблон, совпадения с которым должны остаться в строке.		да
`replace_single`	строка	Строка, на которую будут заменены отдельные символы, не совпадающие с шаблоном.		нет
`replace_repeated`	строка	Строка, на которую будут заменены последовательности нескольких символов, не совпадающих с шаблоном.		нет

Примеры

Очистка строки с помощью регулярного выражения

Пример 862. Исходный код

```
sieve("test123%456.□□□□□.net.", r'[a-z0-9.]')
```

Пример 863. Результат

```
test123456..net.
```

Очистка с указанием заменяющих строк

Пример 864. Исходный код

```
sieve("test123%456.□□□□□.net.", r'[a-z.0-9]', replace_single: "X", replace_repeated:
"<REMOVED>")
```

Пример 865. Результат

```
test123X456.<REMOVED>.net.
```

truncate

Обрезает строку `value` до количества символов `limit`.

Спецификация функции

```
truncate(value: <строка>, limit: <целое число | число с плавающей точкой>, suffix:
<строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для обрезания.		да
`limit`	целое число число с плавающей точкой	Количество символов, после которых будет обрезана строка.		да
`suffix`	строка	Суффикс, который добавляется к обрезанным строкам.		нет

Примеры

Обрезка с многоточием

Пример 866. Исходный код

```
truncate("A rather long sentence.", limit: 11, suffix: "...")
```

Пример 867. Результат

```
A rather lo...
```

Обрезка с произвольным суффиксом

Пример 868. Исходный код

```
truncate("A rather long sentence.", limit: 11, suffix: "[TRUNCATED]")
```

Пример 869. Результат

```
A rather lo[TRUNCATED]
```

Функции изменения регистра

downcase

Преобразует строку `value` в нижний регистр, где "нижний регистр" определяется по свойству Unicode `Lowercase`.

Спецификация функции

```
downcase(value: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую необходимо преобразовать к нижнему регистру.		да

Примеры

Приведение строки к нижнему регистру

Пример 870. Исходный код

```
downcase("Hello, World!")
```

Пример 871. Результат

```
hello, world!
```

upcase

Приводит `value` к верхнему регистру, который определяется согласно свойству Unicode `Uppercase`.

Спецификация функции

```
upcase(value: <строка>)
```

```
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую нужно привести к верхнему регистру		да

Примеры

Приведение строки к верхнему регистру

Пример 872. Исходный код

```
upcase("Hello, World!")
```

Пример 873. Результат

```
HELLO, WORLD!
```

camelcase

Приводит строку `value` к camelCase ("верблюжьему" регистру). Опционально можно указать исходный регистр, иначе будет произведена попытка определить регистр автоматически.

Спецификация функции

```
camelcase(value: <строка>, [original_case: <строка> ])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую нужно привести к camelCase.		да
`original_case`	строка	Опциональная подсказка об исходном типе регистра. Допустимые значения: • `camelCase`; • `kebab-case`; • `PascalCase`; • `SCREAMING_SNAKE`; • `snake_case`.		нет

Примеры

Приведение строки к camelCase

Пример 874. Исходный код

```
camelcase("input-string")
```

Пример 875. Результат

```
inputString
```

Приведение строки к camelCase с заданным исходным регистром

Пример 876. Исходный код

```
camelcase("input-string", "kebab-case")
```

Пример 877. Результат

```
inputString
```

kebabcase

Приводит строку `value` к kebab-case (регистру "шашлыка"). Опционально можно указать исходный регистр, иначе будет произведена попытка определить регистр автоматически.

Спецификация функции

```
kebabcase(value: <строка>, [original_case: <строка> ])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую нужно привести к kebab-case.		да
`original_case`	строка	Опциональная подсказка об исходном типе регистра. Допустимые значения: • `camelCase`; • `kebab-case`; • `PascalCase`; • `SCREAMING_SNAKE`; • `snake_case`.		нет

Примеры

Приведение строки к kebab-case

Пример 878. Исходный код

```
kebabcase("InputString")
```

Пример 879. Результат

```
input-string
```

Приведение строки к kebab-case с заданным исходным регистром

Пример 880. Исходный код

```
kebabcase("InputString", "PascalCase")
```

Пример 881. Результат

```
input-string
```

screamingsnakecase

Приводит строку `value` к SCREAMING_SNAKE case (регистру "кричащей змеи"). Опционально можно указать исходный регистр, иначе будет произведена попытка определить регистр автоматически.

Спецификация функции

```
screamingsnakecase(value: <строка>, [original_case: <строка> ])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую нужно привести к SCREAMING_SNAKE case.		да
`original_case`	строка	Опциональная подсказка об исходном типе регистра. Допустимые значения: • `camelCase`; • `kebab-case`; • `PascalCase`; • `SCREAMING_SNAKE`; • `snake_case`.		нет

Примеры

Приведение строки к SCREAMING_SNAKE case

Пример 882. Исходный код

```
screamingsnakecase("input-string")
```

Пример 883. Результат

```
INPUT_STRING
```

Приведение строки к SCREAMING_SNAKE case с заданным исходным регистром

Пример 884. Исходный код

```
screamingsnakecase("input-string", "kebab-case")
```

Пример 885. Результат

```
INPUT_STRING
```

snakecase

Приводит строку `value` к snake_case ("змеиному" регистру). Опционально можно указать исходный регистр, иначе будет произведена попытка определить регистр автоматически.

Спецификация функции

```
snakecase(value: <строка>, [original_case: <строка> ])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую нужно привести к snake_case.		да
`original_case`	строка	Опциональная подсказка об исходном типе регистра. Допустимые значения: • `camelCase`; • `kebab-case`; • `PascalCase`; • `SCREAMING_SNAKE`; • `snake_case`.		нет

Примеры

Приведение строки к snake_case

Пример 886. Исходный код

```
snakecase("input-string")
```

Пример 887. Результат

```
input_string
```

Приведение строки к snake_case с заданным исходным регистром

Пример 888. Исходный код

```
snakecase("input-string", "kebab-case")
```

Пример 889. Результат

```
input_string
```

pascalcase

Приводит строку `value` к PascalCase (регистру Pascal). Опционально можно указать исходный регистр, иначе будет произведена попытка определить регистр автоматически.

Спецификация функции

```
pascalcase(value: <строка>, [original_case: <строка> ])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, которую нужно привести к PascalCase.		да
`original_case`	строка	Опциональная подсказка об исходном типе регистра. Допустимые значения: • `camelCase`; • `kebab-case`; • `PascalCase`; • `SCREAMING_SNAKE`; • `snake_case`.		нет

Примеры

Приведение строки к PascalCase

Пример 890. Исходный код

```
pascalcase("input-string")
```

Пример 891. Результат

```
InputString
```

Приведение строки к PascalCase с заданным исходным регистром

Пример 892. Исходный код

```
pascalcase("input-string", "kebab-case")
```

Пример 893. Результат

```
InputString
```

Функции типов и форматирования

В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции проверки типов

is_null

Проверяет, является ли значение `value` пустым (`null`) или нет.

Для более гибкой проверки используйте функцию `is_nullish`.

Спецификация функции

```
is_null(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить		да

Примеры

Значение `null`

Пример 894. Исходный код

```
is_null(null)
```

Пример 895. Результат

```
true
```

Несоответствующий тип

Пример 896. Исходный код

```
is_null("строка")
```

Пример 897. Результат

```
false
```

is_nullish

Определяет, является ли `value` "nullish", то есть не имеющим значимого содержания. "Nullish" считаются значения:

- `null`;
- `"-"`;
- состоящие только из пробелов, то есть символов со свойством Unicode `White_Space`.

Спецификация функции

```
is_nullish(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить на отсутствие значимого содержания		да

Примеры

Обнаружение `null` (пустая строка)

Пример 898. Исходный код

```
is_nullish("")
```

Пример 899. Результат

```
true
```

Обнаружение `null` (строка с дефисом)

Пример 900. Исходный код

```
is_nullish("-")
```

Пример 901. Результат

```
true
```

Обнаружение `null` (пустая строка)

Пример 902. Исходный код

```
is_nullish("


")
```

Пример 903. Результат

```
true
```

is_empty

Проверяет, имеет ли объект, массив или строка нулевую длину.

Спецификация функции

```
is_empty(value: <объект | массив | строка>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	объект массив строка	Значение, которое необходимо проверить.		да

Примеры

Пустой массив

Пример 904. Исходный код

```
is_empty([])
```

Пример 905. Результат

```
true
```

Непустая строка

Пример 906. Исходный код

```
is_empty("строка")
```

Пример 907. Результат

```
false
```

Непустой объект

Пример 908. Исходный код

```
is_empty({"foo": "bar"})
```

Пример 909. Результат

```
false
```

is_boolean

Проверяет, является ли `value` логическим значением.

Спецификация функции

```
is_boolean(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любое	Значение, которое необходимо проверить		да

Примеры

Допустимое логическое значение

Пример 910. Исходный код

```
is_boolean(false)
```

Пример 911. Результат

```
true
```

Несоответствующий тип

Пример 912. Исходный код

```
is_boolean("строка")
```

Пример 913. Результат

```
false
```

is_integer

Проверяет, является ли `value` целым числом.

Спецификация функции

```
is_integer(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить.		да

Примеры

Допустимое целое число

Пример 914. Исходный код

```
is_integer(1)
```

Пример 915. Результат

```
true
```

Несоответствующий тип

Пример 916. Исходный код

```
is_integer("строка")
```

Пример 917. Результат

```
false
```

is_float

Проверяет, является ли `value` числом с плавающей точкой.

Спецификация функции

```
is_float(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить.		да

Примеры

Допустимое число с плавающей точкой

Пример 918. Исходный код

```
is_float(0.577)
```

Пример 919. Результат

```
true
```

Несоответствующий тип

Пример 920. Исходный код

```
is_float("строка")
```

Пример 921. Результат

```
false
```

is_timestamp

Проверяет, является ли `value` меткой времени (timestamp).

Спецификация функции

```
is_timestamp(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить.		да

Примеры

Допустимая метка времени (timestamp)

Пример 922. Исходный код

```
is_timestamp(t'2021-03-26T16:00:00Z')
```

Пример 923. Результат

```
true
```

Несоответствующий тип

Пример 924. Исходный код

```
is_timestamp("строка")
```

Пример 925. Результат

```
false
```

is_string

Проверяет, является ли `value` строкой.

Спецификация функции

```
is_string(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить		да

Примеры

Допустимая строка

Пример 926. Исходный код

```
is_string("строка")
```

Пример 927. Результат

```
true
```

Несоответствующий тип

Пример 928. Исходный код

```
is_string([1, 2, 3])
```

Пример 929. Результат

```
false
```

rv_is_utf8

Проверяет, является ли `value` строкой, закодированной в UTF-8.

Спецификация функции

```
rv_is_utf8(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить		да

Примеры

Допустимая строка

Пример 930. Исходный код

```
rv_is_utf8("valid_string")
```

Пример 931. Результат

```
true
```

Недопустимая строка

Пример 932. Исходный код

```
rv_is_utf8(b"\xff123123")
```

Пример 933. Результат

```
false
```

Не строка

Пример 934. Исходный код

```
rv_is_utf8(123)
```

Пример 935. Результат

```
false
```

is_ipv4

Проверяет, является ли `value` допустимым IPv4-адресом или нет.

IPv4-совместимые и IPv4-преобразованные IPv6-адреса не считаются допустимыми.

Спецификация функции

```
is_ipv4(value: <строка>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	IP-адрес, который необходимо проверить.		да

Примеры

Допустимый IPv4-адрес

Пример 936. Исходный код

```
is_ipv4("10.0.102.37")
```

Пример 937. Результат

```
true
```

Допустимый IPv6-адрес

Пример 938. Исходный код

```
is_ipv4("2001:0db8:85a3:0000:0000:8a2e:0370:7334")
```

Пример 939. Результат

```
false
```

Произвольная строка

Пример 940. Исходный код

```
is_ipv4("foobar")
```

Пример 941. Результат

```
false
```

is_ipv6

Проверяет, является ли `value` допустимым IPv6-адресом или нет.

Спецификация функции

```
is_ipv6(value: <строка>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	IP-адрес, который необходимо проверить.		да

Примеры

Допустимый IPv6-адрес

Пример 942. Исходный код

```
is_ipv6("2001:0db8:85a3:0000:0000:8a2e:0370:7334")
```

Пример 943. Результат

```
true
```

Допустимый IPv4-адрес

Пример 944. Исходный код

```
is_ipv6("10.0.102.37")
```

Пример 945. Результат

```
false
```

Произвольная строка

Пример 946. Исходный код

```
is_ipv6("foobar")
```

Пример 947. Результат

```
false
```

is_json

Проверяет, является ли `value` допустимым документом JSON.

Спецификация функции

```
is_json(value: <строка>, [variant: <строка>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Значение, которое необходимо проверить		да
`variant`	строка	Вариант типа JSON для явной проверки: • `object` — объект JSON; • `array` — массив JSON; • `string` — строковое значение в формате JSON, заключенное в кавычки; • `number` — целое число или число с плавающей запятой; • `bool` — `true` или `false`; • `null` — значение `null`.		нет

Примеры

Допустимый JSON-объект

Пример 948. Исходный код

```
is_json("{}")
```

Пример 949. Результат

```
true
```

Недопустимое значение

Пример 950. Исходный код

```
is_json("{")
```

Пример 951. Результат

```
false
```

Точный вариант

Пример 952. Исходный код

```
is_json("{}", variant: "object")
```

Пример 953. Результат

```
true
```

Недопустимый точный вариант

Пример 954. Исходный код

```
is_json("{}", variant: "array")
```

Пример 955. Результат

```
false
```

is_regex

Проверяет, является ли `value` регулярным выражением.

Спецификация функции

```
is_regex(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить		да

Примеры

Допустимое регулярное выражение

Пример 956. Исходный код

```
is_regex(r'pattern')
```

Пример 957. Результат

```
true
```

Несоответствующий тип

Пример 958. Исходный код

```
is_regex("строка")
```

Пример 959. Результат

```
false
```

is_array

Проверяет, является ли `value` массивом.

Спецификация функции

```
is_array(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить		да

Примеры

Допустимый массив

Пример 960. Исходный код

```
is_array([1, 2, 3])
```

Пример 961. Результат

```
true
```

Несоответствующий тип

Пример 962. Исходный код

```
is_array("строка")
```

Пример 963. Результат

```
false
```

is_object

Проверяет, является ли `value` объектом.

Спецификация функции

```
is_object(value: <любой тип>)
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, которое необходимо проверить		да

Примеры

Допустимый объект

Пример 964. Исходный код

```
is_object({"foo": "bar"})
```

Пример 965. Результат

```
true
```

Несоответствующий тип

Пример 966. Исходный код

```
is_object("строка")
```

Пример 967. Результат

```
false
```

tag_types_externally

Добавляет информацию о типе ко всем вложенным скалярным значениям в предоставленном значении `value`.

Информация о типе добавляется в ключ, то есть после преобразования `value` имеет форму объекта с полями `"тип": значение`.

Спецификация функции

```
tag_types_externally(value: <любой тип>)
:: <объект | массив объектов | пустое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любое	Значение, которое нужно пометить типами.		да

Примеры

Пометить тип в скалярном значении

Пример 968. Исходный код

```
tag_types_externally(123)
```

Пример 969. Результат

```
{
  "integer": 123
}
```

Пометить типы в объекте

Пример 970. Исходный код

```
tag_types_externally({
  "message": "Hello world",
  "request": {
    "duration_ms": 67.9
  }
})
```

Пример 971. Результат

```
{
  "message": {
    "string": "Hello world"
  },
  "request": {
    "duration_ms": {
      "float": 67.9
    }
  }
}
```

Пометить типы в массиве

Пример 972. Исходный код

```
tag_types_externally(["foo", "bar"])
```

Пример 973. Результат

```
[
  {
    "string": "foo"
  },
  {
    "string": "bar"
  }
]
```

Типы в пустом значении

Пример 974. Исходный код

```
tag_types_externally(null)
```

Пример 975. Результат

```
null
```

Функции валидации типов

bool

Возвращает `value`, если это логическое значение, иначе возвращает ошибку. Это позволяет проверяющему типу гарантировать, что возвращаемое значение является логическим и может быть использовано в любой функции, которая ожидает логическое значение.

Спецификация функции

```
bool(value: <любой тип>)
:: <логическое значение> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любое	Значение, которое необходимо проверить на принадлежность к логическим.		да

Ошибки

Функция `bool` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является логическим значением.

Примеры

Объявление типа Boolean

Пример 976. Исходный код

```
bool!(.value)
```

Пример 977. Результат

```
false
```

int

Возвращает `value`, если оно является целым числом, иначе возвращает ошибку. Это позволяет проверке типов гарантировать, что возвращаемое значение является целым числом и может быть использовано в любой функции, ожидающей такой тип данных.

Спецификация функции

```
int(value: <любой тип>)
:: <целое число> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любой	Значение, для которого необходимо убедиться, что оно является целым числом.		да

Ошибки

Функция `int` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является целым числом.

Примеры

Объявление переменной типа целое число

Пример 978. Исходный код

```
int!(.value)
```

Пример 979. Результат

```
42
```

float

Возвращает `value`, если это число с плавающей точкой, и ошибки в противном случае. Это позволяет средству проверки типов гарантировать, что возвращаемое значение является числом с плавающей точкой и может использоваться в любой функции, которая его ожидает.

Спецификация функции

```
float(value: <любое>)
:: <число с плавающей точкой>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любое	Значение, которое необходимо проверить на тип float.		да

Ошибки

Функция `float` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является числом с плавающей точкой.

Примеры

Объявление типа float

Пример 980. Исходный код

```
float!(.value)
```

Пример 981. Результат

```
42
```

timestamp

Возвращает `value`, если это метка времени, иначе возвращает ошибку. Это позволяет проверке типов гарантировать, что возвращенное значение является меткой времени и может быть использовано в любой функции, которая ожидает метку времени.

Спецификация функции

```
timestamp(value: <любой тип>)
:: <метка времени> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любое	Значение, которое нужно обеспечить меткой времени.		да

Ошибки

Функция `timestamp` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является меткой времени.

Примеры

Объявление типа метки времени

Пример 982. Исходный код

```
ok, err = timestamp(.timestamp)
```

Пример 983. Результат

```
function call error for "timestamp" at (10:31): expected timestamp, got string
```

string

Возвращает `value`, если оно является строкой, иначе генерирует ошибку. Это гарантирует, что возвращаемое значение является строкой, которая может быть использована в любой функции, ожидающей строку.

Спецификация функции

```
string(value: <любой тип>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любое	Значение, для которого нужно гарантировать тип строки.		да

Ошибки

Функция `string` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является строкой.

Примеры

Объявление типа строки

Пример 984. Исходный код

```
string!(.message)
```

Пример 985. Результат

```
{"поле": "значение"}
```

array

Возвращает `value`, если оно является массивом, иначе возвращает ошибку. Это позволяет проверить тип и гарантирует, что возвращаемое значение — это массив, который может быть использован в любой функции, ожидающей массив.

Спецификация функции

```
array(value: <любой тип>)
:: <массив> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любое	Значение, проверяемое на предмет того, является ли оно массивом.		да

Ошибки

Функция `array` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является массивом.

Примеры

Объявление массива

Пример 986. Исходный код

```
array!(.value)
```

Пример 987. Результат

```
[
  1,
  2,
  3
]
```

object

Возвращает `value`, если оно является объектом, иначе возвращает ошибку. Это позволяет проверить, что возвращаемое значение является объектом и может быть использовано в любой функции, ожидающей объект.

Спецификация функции

```
object(value: <любой тип>)
:: <объект>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любое	Значение, которое требуется проверить		да

Ошибки

Функция `object` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является объектом.

Примеры

Определение типа объекта

Пример 988. Исходный код

```
object!(.value)
```

Пример 989. Результат

```
{
  "field1": "value1",
  "field2": "value2"
}
```

Функции представлений чисел

format_number

Форматирует `value` в строковое представление числа.

Спецификация функции

```
format_number(value: <целое число | число с плавающей точкой>, [scale: <целое число>,
decimal_separator: <строка>, grouping_separator: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой	Число для форматирования в строку.		да
`scale`	целое число	Количество десятичных знаков для отображения.		нет
`decimal_separator`	строка	Символ, используемый между целой и десятичной частями числа.	`.`	нет
`grouping_separator`	строка	Символ, используемый между каждой тысячной частью числа.	`,`	нет

Примеры

Форматирование числа (3 десятичных знака)

Пример 990. Исходный код

```
format_number(1234567.89, 3, decimal_separator: ".", grouping_separator: ",")
```

Пример 991. Результат

```
"1,234,567.890"
```

format_int

Форматирует целое число `value` в строковое представление с использованием заданного основания.

Спецификация функции

```
format_int(value: <целое число>, [base: <целое число>])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число	Число для форматирования.		да
`base`	целое число	Основание для форматирования числа. Должно быть между 2 и 36 включительно.	`10`	нет

Ошибки

Функция `format_int` **может возвращать ошибки**, для которых требуется обработка:

- Основание не находится между 2 и 36.

Примеры

Форматирование как шестнадцатеричное число

Пример 992. Исходный код

```
format_int!(42, 16)
```

Пример 993. Результат

```
"2a"
```

Форматирование как отрицательное шестнадцатеричное число

Пример 994. Исходный код

```
format_int!(-42, 16)
```

Пример 995. Результат

```
"-2a"
```

parse_int

Разбирает строку `value`, представляющую число в указанной системе счисления, в целое число.

Спецификация функции

```
parse_int(value: <строка>, [base: <целое число>])
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`base`	целое число	Система счисления числа. Должна быть в диапазоне от 2 до 36 включительно. Если не указано, для определения системы счисления будет использоваться префикс строки: * `0b` — двоичная; * `0`, `0o` — восьмеричная; * `0x` — шестнадцатеричная; * в остальных случаях — десятеричная.		нет

Ошибки

Функция `parse_int` **может возвращать ошибки**, для которых требуется обработка:

- Значение `base` не находится в диапазоне от 2 до 36;
- Число не может быть разобрано в указанной системе счисления.

Примеры

Разбор десятичного числа

Пример 996. Исходный код

```
parse_int!("-42")
```

Пример 997. Результат

```
-42
```

Разбор двоичного числа

Пример 998. Исходный код

```
parse_int!("0b1001")
```

Пример 999. Результат

```
9
```

Разбор восьмеричного числа

Пример 1000. Исходный код

```
parse_int!("0o42")
```

Пример 1001. Результат

```
34
```

Разбор шестнадцатеричного числа

Пример 1002. Исходный код

```
parse_int!("0x2a")
```

Пример 1003. Результат

```
42
```

Разбор числа с явно указанной системой счисления

Пример 1004. Исходный код

```
parse_int!("2a", 17)
```

Пример 1005. Результат

```
44
```

parse_float

Разбирает строку `value`, представляющую число в десятичной системе счисления, в число с плавающей запятой.

Спецификация функции

```
parse_float(value: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для разбора.		да

Ошибки

Функция `parse_float` **может возвращать ошибки**, для которых требуется обработка:

- `value` не может быть разобрано как число.
- Значение `NaN` не поддерживается.

Примеры

Разбор отрицательного числа

Пример 1006. Исходный код

```
parse_float!("-42")
```

Пример 1007. Результат

```
-42.0
```

Разбор числа с плавающей запятой

Пример 1008. Исходный код

```
parse_float!("42.38")
```

Пример 1009. Результат

```
42.38
```

Разбор экспоненциальной записи

Пример 1010. Исходный код

```
parse_float!("2.5e3")
```

Пример 1011. Результат

```
2500.0
```

to_int

Приводит `value` к целому числу.

Спецификация функции

```
to_int(value: <целое число | число с плавающей точкой | логическое значение | строка |
метка времени>)
:: <целое число> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой логическое значение строка метка времени	Значение для преобразования в целое число.		да

Характеристики функции

Поведение функции зависит от типа `value`:

- При преобразовании чисел с плавающей точкой отбрасывается дробная часть.
- При преобразовании логических значений `true` преобразуется в `1`, `false` в `0`.
- Строки преобразуются, только если состоят из десятичных чисел и знаков.
- Метки времени преобразуются в формат времени Unix.

Ошибки

Функция `to_int` **может возвращать ошибки**, для которых требуется обработка:

- `value` является строкой, но текст не является целым числом
- `value` не является строкой, числом или меткой времени

Примеры

Приведение строки к целому числу

Пример 1012. Исходный код

```
to_int!("-02")
```

Пример 1013. Результат

```
2
```

Приведение метки времени к целому числу

Пример 1014. Исходный код

```
to_int(t'2020-12-30T22:20:53.824727Z')
```

Пример 1015. Результат

```
1609366853
```

Приведение числа с плавающей точкой к целому числу

Пример 1016. Исходный код

```
to_int(2.6)
```

Пример 1017. Результат

```
2
```

to_float

Приводит `value` к типу с плавающей точкой.

Спецификация функции

```
to_float(value: <целое число | число с плавающей точкой | логическое значение | строка
| метка времени>)
:: <число с плавающей точкой>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой логическое значение строка метка времени	Значение для преобразования в число с плавающей точкой. Должно быть преобразуемым в число с плавающей точкой, иначе будет сгенерирована ошибка.		да

Характеристики функции

Поведение функции зависит от типа `value`:

- При преобразовании логических значений `true` принимает значение 1,`false`-- 0.
- Строки преобразуются, только если состоят из десятичных чисел, знаков и десятичной точки.
- Метки времени преобразуются в формат времени Unix с указанием наносекунд.

Ошибки

Функция `to_float` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является поддерживаемым представлением числа с плавающей точкой.
- `value`, представленное меткой времени, находится вне допустимого диапазона для меток времени.

Примеры

Приведение к числу с плавающей точкой

Пример 1018. Исходный код

```
to_float!("3.145")
```

Пример 1019. Результат

```
3.145
```

Приведение метки времени к числу с плавающей точкой

Пример 1020. Исходный код

```
to_float(t'2020-12-30T22:20:53.824727Z')
```

Пример 1021. Результат

```
1609366853.824727
```

Приведение строки к числу с плавающей точкой

Пример 1022. Исходный код

```
to_float!(".5")
```

Пример 1023. Результат

```
0.5
```

Функции преобразования чисел

abs

Вычисляет модуль числа `value`.

Спецификация функции

```
abs(value: <целое число | число с плавающей точкой>)
:: <целое число | число с плавающей точкой>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой	Число для вычисления модуля.		да

Примеры

Вычисление модуля целого числа

Пример 1024. Исходный код

```
abs(-42)
```

Пример 1025. Результат

```
42
```

Вычисление модуля числа с плавающей точкой

Пример 1026. Исходный код

```
abs(-42.2)
```

Пример 1027. Результат

```
42.2
```

ceil

Округляет `value` к большему до указанной точности `precision`.

Спецификация функции

```
ceil(value: <целое число | число с плавающей точкой>, [precision: <целое число>])
:: <целое число | число с плавающей точкой>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой	Число, которое нужно округлить вверх.		да
`precision`	целое число	Количество десятичных знаков для округления.	`0`	нет

Примеры

Округление числа к большему (без указания точности)

Пример 1028. Исходный код

```
ceil(4.345)
```

Пример 1029. Результат

```
5
```

Округление числа к большему (с указанием точности)

Пример 1030. Исходный код

```
ceil(4.345, precision: 2)
```

Пример 1031. Результат

```
4.35
```

floor

Округляет `value` вниз до указанной `precision` (точности).

Спецификация функции

```
floor(value: <целое число | число с плавающей точкой>, [precision: <целое число>])
:: <целое число | число с плавающей точкой>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой	Число, которое нужно округлить вниз.		да
`precision`	целое число	Количество десятичных знаков для округления.	`0`	нет

Примеры

Округление числа вниз (без указания точности)

Пример 1032. Исходный код

```
floor(4.345)
```

Пример 1033. Результат

```
4
```

Округление числа вниз (с указанием точности)

Пример 1034. Исходный код

```
floor(4.345, precision: 2)
```

Пример 1035. Результат

```
4.34
```

round

Округляет `value` до указанной точности `precision`.

Спецификация функции

```
round(value: <целое число | число с плавающей точкой>, [precision: <целое число>])
:: <целое число | число с плавающей точкой>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой	Число для округления.		да
`precision`	целое число	Количество знаков после запятой для округления.	`0`	нет

Примеры

Округление числа (без указания точности)

Пример 1036. Исходный код

```
round(4.345)
```

Пример 1037. Результат

```
4
```

Округление числа (с указанием точности)

Пример 1038. Исходный код

```
round(4.345, precision: 2)
```

Пример 1039. Результат

```
4.35
```

mod

Вычисляет остаток от деления `value` на `modulus`.

Спецификация функции

```
mod(value: <целое число | число с плавающей точкой>, modulus: <целое число | число с
плавающей точкой>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой	Значение, для которого применяется деление по модулю.		да
`modulus`	целое число число с плавающей точкой	Значение модуля.		да

Ошибки

Функция `mod` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является целым числом или числом с плавающей точкой.
- `modulus` не является целым числом или числом с плавающей точкой.
- `modulus` равен 0.

Примеры

Вычисление остатка от деления двух целых чисел

Пример 1040. Исходный код

```
remainder = mod(5, 2)
```

Пример 1041. Результат

```
1
```

Функции форматирования меток времени

from_unix_timestamp

Преобразует целое число `value` из формата времени Unix в формат времени VRL (`timestamp`).

Преобразует из количества секунд с начала эпохи Unix по умолчанию, но также можно указать миллисекунды или наносекунды с помощью параметра `unit`.

Спецификация функции

```
from_unix_timestamp(value: <целое число>, [unit: <строка>])
:: <метка времени>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число	Целое число Unix-времени для преобразования		да
`unit`	строка	Единица измерения времени: • `seconds`; • `milliseconds`; • `microseconds`; • `nanoseconds`.	`seconds`	нет

Ошибки

Функция `from_unix_timestamp` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является целым числом.
- В аргументе `unit` указана неизвестная единица.

Примеры

Преобразование из формата Unix-времени (секунды)

Пример 1042. Исходный код

```
from_unix_timestamp!(5)
```

Пример 1043. Результат

```
1970-01-01T00:00:05Z
```

Преобразование из формата Unix-времени (миллисекунды)

Пример 1044. Исходный код

```
from_unix_timestamp!(5000, unit: "milliseconds")
```

Пример 1045. Результат

```
1970-01-01T00:00:05Z
```

Преобразование из формата Unix-времени (наносекунды)

Пример 1046. Исходный код

```
from_unix_timestamp!(5000, unit: "nanoseconds")
```

Пример 1047. Результат

```
1970-01-01T00:00:00.000005Z
```

to_unix_timestamp

Преобразует значение метки времени (`value`) в метку времени Unix.

По умолчанию возвращает количество секунд с момента начала эпохи Unix, но можно также указать миллисекунды микросекунды или наносекунды в аргументе `unit`.

Спецификация функции

```
to_unix_timestamp(value: <метка времени>, [unit: <строка>])
:: <целое число>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	метка времени	Метка времени для преобразования в Unix.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`unit`	строка	Единица измерения времени: • `seconds`; • `milliseconds`; • `microseconds`; • `nanoseconds`.	`seconds`	нет

Ошибки

Функция `to_unix_timestamp` **может возвращать ошибки**, для которых требуется обработка:

- `value` вне допустимого диапазона для меток времени.
- В аргументе `unit` указана неизвестная единица.

Примеры

Преобразование в метку времени Unix (секунды)

Пример 1048. Исходный код

```
to_unix_timestamp(t'2021-01-01T00:00:00+00:00')
```

Пример 1049. Результат

```
1609459200
```

Преобразование в метку времени Unix (миллисекунды)

Пример 1050. Исходный код

```
to_unix_timestamp(t'2021-01-01T00:00:00Z', unit: "milliseconds")
```

Пример 1051. Результат

```
1609459200000
```

Преобразование в метку времени Unix (наносекунды)

Пример 1052. Исходный код

```
to_unix_timestamp(t'2021-01-01T00:00:00Z', unit: "nanoseconds")
```

Пример 1053. Результат

```
1609459200000000000
```

format_timestamp

Форматирует `value` в строковое представление времени.

Спецификация функции

```
format_timestamp(value: <метка времени>, format: <строка>, [timezone: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	метка времени	Метка времени для форматирования в текст.		да
`format`	строка	Строка формата, описанная в библиотеке Chrono.		да
`timezone`	строка	Часовой пояс, используемый при форматировании метки времени. Используйте идентификатор TZ или 'local'.		нет

Примеры

Форматирование метки времени (ISO8601/RFC 3339)

Пример 1054. Исходный код

```
format_timestamp!(t'2020-10-21T16:00:00Z', format: "%+")
```

Пример 1055. Результат

```
2020-10-21T16:00:00+00:00
```

Форматирование метки времени (пользовательский формат)

Пример 1056. Исходный код

```
format_timestamp!(t'2020-10-21T16:00:00Z', format: "%v %R")
```

Пример 1057. Результат

```
21-Oct-2020 16:00
```

parse_timestamp

Разбирает `value` в формате strptime.

```
По умолчанию функция использует системный часовой пояс для разбора меток времени. Для
анализа временных меток без указания часового пояса, таких как 16/10/2019 12:00:00,
можно использовать аргумент `timezone`.
```

Спецификация функции

```
parse_timestamp(value: <строка>, format: <строка>, [timezone: <строка>])
:: <метка времени> , <ошибка>
```

Аргумент	Тип	Описание	Обязателен
`value`	строка	Текст времени.	да
`format`	строка	Формат strptime.	да
`timezone`	строка	Часовой пояс в формате tz database.	нет

Ошибки

Функция `parse_timestamp` **может возвращать ошибки**, для которых требуется обработка:

- `value` не удалось разобрать с использованием предоставленного `format`;
- `value` не удалось разобрать с использованием предоставленного `timezone`.

Примеры

Разбор метки времени

Пример 1058. Исходный код

```
parse_timestamp!("10-Oct-2020 16:00+00:00", format: "%v %R %:z")
```

Пример 1059. Результат

```
2020-10-10T16:00:00Z
```

Разбор метки времени с указанием часового пояса

Пример 1060. Исходный код

```
parse_timestamp!("16/10/2019 12:00:00", format: "%d/%m/%Y %H:%M:%S", timezone:
"Asia/Taipei")
```

Пример 1061. Результат

```
2019-10-16T04:00:00Z
```

Функции приведения строк и логических значений

to_bool

Приводит `value` к логическому значению.

Спецификация функции

```
to_bool(value: <логическое значение | целое число | число с плавающей точкой | пустое
значение | строка>)
:: <логическое значение> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	логическое значение целое число число с плавающей точкой пустое значение строка	Значение для преобразования в логическое значение.		да

Ошибки

Функция `to_bool` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является поддерживаемым представлением логического значения.

Примеры

Приведение к логическому значению (строка)

Пример 1062. Исходный код

```
to_bool!("yes")
```

Пример 1063. Результат

```
true
```

Приведение к логическому значению (число с плавающей точкой)

Пример 1064. Исходный код

```
to_bool(0.0)
```

Пример 1065. Результат

```
false
```

Приведение к логическому значению (целое число)

Пример 1066. Исходный код

```
to_bool(0)
```

Пример 1067. Результат

```
false
```

Приведение к логическому значению (null)

Пример 1068. Исходный код

```
to_bool(null)
```

Пример 1069. Результат

```
false
```

Приведение к логическому значению (логическое значение)

Пример 1070. Исходный код

```
to_bool(true)
```

Пример 1071. Результат

```
true
```

to_regex

Приводит `value` к регулярному выражению.

Спецификация функции

```
to_regex(value: <строка>)
:: <регулярное выражение> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Значение для преобразования в регулярное выражение		да

Примечания

У этой функции есть особое поведение, о котором стоит знать.

- Компиляция регулярного выражения — это ресурсозатратная операция и может ограничивать пропускную способность Vector. Используйте эту функцию, только если вы абсолютно уверены, что нет другого способа.

Ошибки

Функция `to_regex` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является строкой.

Примеры

Приведение к регулярному выражению

Пример 1072. Исходный код

```
to_regex("^foo$") ?? r''
```

Пример 1073. Результат

```
^foo$
```

to_string

Приводит `value` к строке.

Спецификация функции

```
to_string(value: < целое число | число с плавающей точкой | логическое значение |
строка | метка времени>)
:: <число с плавающей точкой | пустое значение>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	целое число число с плавающей точкой логическое значение строка метка времени пустое значение	Значение для преобразования в строку		да

Ошибки

Функция `to_string` **может возвращать ошибки**, для которых требуется обработка:

- `value` не является целым числом, числом с плавающей точкой, логическим значением, строкой, меткой времени или пустым значением

Примеры

Приведение к строке (логическое значение)

Пример 1074. Исходный код

```
to_string(true)
```

Пример 1075. Результат

```
"true"
```

Приведение к строке (целое число)

Пример 1076. Исходный код

```
to_string(52)
```

Пример 1077. Результат

```
"52"
```

Приведение к строке (число с плавающей точкой)

Пример 1078. Исходный код

```
to_string(52.2)
```

Пример 1079. Результат

```
"52.2"
```

Хэш-функции и криптографические функции

В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции хэширования и контрольных сумм

crc

Вычисляет CRC (контрольную сумму) значения. Используемый алгоритм вычисления задается опционально.

Спецификация функции

```
crc(value: <строка>, [algorithm: <строка>])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, для которой вычисляется контрольная сумма.		да
`algorithm`	строка	Алгоритм CRC. Допустимые значения см. ниже.	`"CRC_32_ISO_HDLC"`	нет

▼ Допустимые алгоритмы CRC:

- `"CRC_3_GSM"`;
- `"CRC_3_ROHC"`;
- `"CRC_4_G_704"`;
- `"CRC_4_INTERLAKEN"`;
- `"CRC_5_EPC_C1G2"`;
- `"CRC_5_G_704"`;
- `"CRC_5_USB"`;
- `"CRC_6_CDMA2000_A"`;
- `"CRC_6_CDMA2000_B"`;

- "CRC_6_DARC";
- "CRC_6_GSM";
- "CRC_6_G_704";
- "CRC_7_MMC";
- "CRC_7_ROHC";
- "CRC_7_UMTS";
- "CRC_8_AUTOSAR";
- "CRC_8_BLUETOOTH";
- "CRC_8_CDMA2000";
- "CRC_8_DARC";
- "CRC_8_DVB_S2";
- "CRC_8_GSM_A";
- "CRC_8_GSM_B";
- "CRC_8_HITAG";
- "CRC_8_I_432_1";
- "CRC_8_I_CODE";
- "CRC_8_LTE";
- "CRC_8_MAXIM_DOW";
- "CRC_8_MIFARE_MAD";
- "CRC_8_NRSC_5";
- "CRC_8_OPENSAFETY";
- "CRC_8_ROHC";
- "CRC_8_SAE_J1850";
- "CRC_8_SMBUS";
- "CRC_8_TECH_3250";
- "CRC_8_WCDMA";
- "CRC_10_ATM";
- "CRC_10_CDMA2000";
- "CRC_10_GSM";
- "CRC_11_FLEXRAY";
- "CRC_11_UMTS";
- "CRC_12_CDMA2000";
- "CRC_12_DECT";
- "CRC_12_GSM";
- "CRC_12_UMTS";

- "CRC_13_BBC";
- "CRC_14_DARC";
- "CRC_14_GSM";
- "CRC_15_CAN";
- "CRC_15_MPT1327";
- "CRC_16_ARC";
- "CRC_16_CDMA2000";
- "CRC_16_CMS";
- "CRC_16_DDS_110";
- "CRC_16_DECT_R";
- "CRC_16_DECT_X";
- "CRC_16_DNP";
- "CRC_16_EN_13757";
- "CRC_16_GENIBUS";
- "CRC_16_GSM";
- "CRC_16_IBM_3740";
- "CRC_16_IBM_SDLC";
- "CRC_16_ISO_IEC_14443_3_A";
- "CRC_16_KERMIT";
- "CRC_16_LJ1200";
- "CRC_16_M17";
- "CRC_16_MAXIM_DOW";
- "CRC_16_MCRF4XX";
- "CRC_16_MODBUS";
- "CRC_16_NRSC_5";
- "CRC_16_OPENSAFETY_A";
- "CRC_16_OPENSAFETY_B";
- "CRC_16_PROFIBUS";
- "CRC_16_RIELLO";
- "CRC_16_SPI_FUJITSU";
- "CRC_16_T10_DIF";
- "CRC_16_TELEDISK";
- "CRC_16_TMS37157";
- "CRC_16_UMTS";
- "CRC_16_USB";

- "CRC_16_XMODEM";
- "CRC_17_CAN_FD";
- "CRC_21_CAN_FD";
- "CRC_24_BLE";
- "CRC_24_FLEXRAY_A";
- "CRC_24_FLEXRAY_B";
- "CRC_24_INTERLAKEN";
- "CRC_24_LTE_A";
- "CRC_24_LTE_B";
- "CRC_24_OPENPGP";
- "CRC_24_OS_9";
- "CRC_30_CDMA";
- "CRC_31_PHILIPS";
- "CRC_32_AIXM";
- "CRC_32_AUTOSAR";
- "CRC_32_BASE91_D";
- "CRC_32_BZIP2";
- "CRC_32_CD_ROM_EDC";
- "CRC_32_CKSUM";
- "CRC_32_ISCSI";
- "CRC_32_ISO_HDLC";
- "CRC_32_JAMCRC";
- "CRC_32_MEF";
- "CRC_32_MPEG_2";
- "CRC_32_XFER";
- "CRC_40_GSM";
- "CRC_64_ECMA_182";
- "CRC_64_GO_ISO";
- "CRC_64_MS";
- "CRC_64_REDIS";
- "CRC_64_WE";
- "CRC_64_XZ";
- "CRC_82_DARC";

Ошибки

Функция **может возвращать ошибки**, для которых требуется обработка:

- `value` не является строкой.
- `algorithm` не является поддерживаемым алгоритмом.

Примеры

Вычислить контрольную сумму CRC согласно алгоритму по умолчанию

Пример 1080. Исходный код

```
crc("foo")
```

Пример 1081. Результат

```
"2356372769"
```

Вычислить контрольную сумму CRC согласно алгоритму `CRC_32_CKSUM`

Пример 1082. Исходный код

```
crc("foo", algorithm: "CRC_32_CKSUM")
```

Пример 1083. Результат

```
"4271552933"
```

hmac

Вычисляет HMAC для значения `value` с использованием указанного ключа `key`. Используемый алгоритм хэширования можно указать опционально.

В большинстве случаев полученный байтовый поток следует закодировать в шестнадцатеричную строку, используя функцию `encode_base16`, или в строку в формате Base64, используя функцию `encode_base64`.

Спецификация функции

```
hmac(value: <строка>, key: <строка>, [algorithm: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для вычисления HMAC.		да
`key`	строка	Строка для использования в качестве криптографического ключа.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`algorithm`	строка	Алгоритм хэширования. Допустимые значения: • `"SHA1"`; • `"SHA-224"`; • `"SHA-256"`; • `"SHA-384"`; • `"SHA-512"`.	`SHA-256`	нет

Ошибки

Эта функция не возвращает ошибок, если в аргументе `algorithm` указано значение по умолчанию или строковый литерал с названием алгоритма, распознаваемым и допустимым на этапе компиляции. В противном случае функция будет возвращать ошибки.

Примеры

Вычислить HMAC сообщения (по умолчанию: SHA-256) и закодировать его в строку Base64

Пример 1084. Исходный код

```
encode_base64(hmac("Hello there", "super-secret-key"))
```

Пример 1085. Результат

```
eLGE8YMviv85NPXgISRUZxstBNSU47JQdcXkUWcClmI=
```

Вычислить HMAC сообщения с использованием SHA-224 и закодировать его в шестнадцатеричную строку

Пример 1086. Исходный код

```
encode_base16(hmac("Hello there", "super-secret-key", algorithm: "SHA-224"))
```

Пример 1087. Результат

```
42fccbc2b7d22a143b92f265a8046187558a94d11ddbb30622207e90
```

Вычислить HMAC сообщения с использованием переменного алгоритма хэширования

Пример 1088. Исходный код

```
.hash_algo = "SHA-256"
hmac_bytes, err = hmac("Hello there", "super-secret-key", algorithm: .hash_algo)
if err == null {
```

```
  .hmac = encode_base16(hmac_bytes)
}
```

Пример 1089. Результат

```
78b184f1832f8aff3934f5e0212454671b2d04d494e3b25075c5e45167029662
```

md5

Вычисляет хэш md5 для `value`.

Спецификация функции

```
md5(value: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка для вычисления хэша.		да

Примеры

Создание хэша md5

Пример 1090. Исходный код

```
md5("foo")
```

Пример 1091. Результат

```
acbd18db4cc2f85cedef654fccc4a4d8
```

seahash

Вычисляет хэш Seahash для `value`.

Спецификация функции

```
seahash(value: <строка>)
:: <целое число>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, для которой вычисляется хэш.		да

Примечания

Эта функция имеет особое поведение, о котором следует знать:

- Из-за ограничений в типах данных VRL функция преобразует результат Seahash из беззнакового 64-битного целого числа в знаковый формат. Значения, превышающие максимальное значение знакового 64-битного целого числа, становятся отрицательными.

Примеры

Вычисление хэша Seahash

Пример 1092. Исходный код

```
seahash("foobar")
```

Пример 1093. Результат

```
5348458858952426000
```

Вычисление отрицательного хэша Seahash

Пример 1094. Исходный код

```
seahash("bar")
```

Пример 1095. Результат

```
-2796170501982571500
```

sha1

Вычисляет хэш SHA-1 для `value`.

Спецификация функции

```
sha1(value: <строка>)
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, для которой вычисляется хэш.		да

Примеры

Вычисление хэша SHA-1

Пример 1096. Исходный код

```
sha1("foo")
```

Пример 1097. Результат

```
0beec7b5ea3f0fdbc95d0dd47f3c5bc275da8a33
```

sha2

Вычисляет хэш SHA-2 для `value`.

Спецификация функции

```
sha2(value: <строка>, [variant: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, для которой вычисляется хэш.		да
`variant`	строка	Вариант алгоритма хэширования: • `"SHA-224"`; • `"SHA-256"`; • `"SHA-384"`; • `"SHA-512"`; • `"SHA-512/224"`; • `"SHA-512/256"`.	`SHA-512/256`	нет

Примеры

Вычисление хэша SHA-2

Пример 1098. Исходный код

```
sha2("foo", variant: "SHA-512/224")
```

Пример 1099. Результат

```
d68f258d37d670cfc1ec1001a0394784233f88f056994f9a7e5e99be
```

sha3

Вычисляет хэш SHA-3 для `value`.

Спецификация функции

```
sha3(value: <строка>, [variant: <строка>])
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	строка	Строка, для которой вычисляется хэш.		да
`variant`	строка	Вариант алгоритма хэширования: • `"SHA3-224"`; • `"SHA3-256"`; • `"SHA3-384"`; • `"SHA3-512"`.	`SHA3-512`	нет

Примеры

Вычисление хэша SHA-3

Пример 1100. Исходный код

```
sha3("foo", variant: "SHA3-224")
```

Пример 1101. Результат

```
f4f6779e153c391bbd29c95e72b0708e39d9166c7cea51d1f10ef58a
```

Функции шифрования

encrypt

Шифрует строку с использованием симметричного алгоритма шифрования.

Поддерживаемые алгоритмы:

- AES-256-CFB (ключ = 32 байта, IV = 16 байт)
- AES-192-CFB (ключ = 24 байта, IV = 16 байт)
- AES-128-CFB (ключ = 16 байт, IV = 16 байт)
- AES-256-OFB (ключ = 32 байта, IV = 16 байт)
- AES-192-OFB (ключ = 24 байта, IV = 16 байт)
- AES-128-OFB (ключ = 16 байт, IV = 16 байт)
- AES-128-SIV (ключ = 32 байта, IV = 16 байт)
- AES-256-SIV (ключ = 64 байта, IV = 16 байт)

- AES-256-CTR-LE (ключ = 32 байта, IV = 16 байт)
- AES-192-CTR-LE (ключ = 24 байта, IV = 16 байт)
- AES-128-CTR-LE (ключ = 16 байт, IV = 16 байт)
- AES-256-CTR-BE (ключ = 32 байта, IV = 16 байт)
- AES-192-CTR-BE (ключ = 24 байта, IV = 16 байт)
- AES-128-CTR-BE (ключ = 16 байт, IV = 16 байт)
- AES-256-CBC-PKCS7 (ключ = 32 байта, IV = 16 байт)
- AES-192-CBC-PKCS7 (ключ = 24 байта, IV = 16 байт)
- AES-128-CBC-PKCS7 (ключ = 16 байт, IV = 16 байт)
- AES-256-CBC-ANSIX923 (ключ = 32 байта, IV = 16 байт)
- AES-192-CBC-ANSIX923 (ключ = 24 байта, IV = 16 байт)
- AES-128-CBC-ANSIX923 (ключ = 16 байт, IV = 16 байт)
- AES-256-CBC-ISO7816 (ключ = 32 байта, IV = 16 байт)
- AES-192-CBC-ISO7816 (ключ = 24 байта, IV = 16 байт)
- AES-128-CBC-ISO7816 (ключ = 16 байт, IV = 16 байт)
- AES-256-CBC-ISO10126 (ключ = 32 байта, IV = 16 байт)
- AES-192-CBC-ISO10126 (ключ = 24 байта, IV = 16 байт)
- AES-128-CBC-ISO10126 (ключ = 16 байт, IV = 16 байт)
- CHACHA20-POLY1305 (ключ = 32 байта, IV = 12 байт)
- XCHACHA20-POLY1305 (ключ = 32 байта, IV = 24 байта)
- XSALSA20-POLY1305 (ключ = 32 байта, IV = 24 байта)

Спецификация функции

```
encrypt(plaintext: <строка>, algorithm: <строка>, key: <строка>, iv: <строка>)
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`plaintext`	строка	Строка для шифрования.		да
`algorithm`	строка	Алгоритм, используемый для шифрования.		да
`key`	строка	Ключ для шифрования. Он должен представлять собой сырые (не закодированные) байты. Его длина должна соответствовать указанному алгоритму.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`iv`	строка	Вектор инициализации (IV) для шифрования. Он должен представлять собой сырые (не закодированные) байты IV. Его длина должна соответствовать указанному алгоритму. Для каждого сообщения следует генерировать новый IV. Вы можете использовать `random_bytes` для генерации криптографически безопасного случайного значения.		да

Ошибки

Функция `encrypt` **может возвращать ошибки**, для которых требуется обработка:

- Алгоритм `algorithm` не поддерживается.
- Длина `key` не соответствует требуемому размеру ключа для указанного алгоритма.
- Длина `iv` не соответствует требуемому размеру вектора инициализации для указанного алгоритма.

Примеры

Шифрование значения

Пример 1102. Исходный код

```
plaintext = "super secret message"
iv = "1234567890123456" # обычно следует вызывать random_bytes(16)
key = "16_byte_keyxxxxx"
encrypted_message = encrypt!(plaintext, "AES-128-CBC-PKCS7", key, iv: iv)
encode_base64(encrypted_message)
```

Пример 1103. Результат

```
GBw8Mu00v0Kc38+/PvsVtGgWuUJ+ZNLgF8Opy8ohIYE=
```

decrypt

Дешифрует строку с помощью симметричного алгоритма шифрования.

Поддерживаемые алгоритмы:

- AES-256-CFB (ключ = 32 байта, iv = 16 байт)
- AES-192-CFB (ключ = 24 байта, iv = 16 байт)
- AES-128-CFB (ключ = 16 байт, iv = 16 байт)

- AES-256-OFB (ключ = 32 байта, iv = 16 байт)
- AES-192-OFB (ключ = 24 байта, iv = 16 байт)
- AES-128-OFB (ключ = 16 байт, iv = 16 байт)
- AES-128-SIV (ключ = 32 байта, IV = 16 байт)
- AES-256-SIV (ключ = 64 байта, IV = 16 байт)
- AES-256-CTR-LE (ключ = 32 байта, IV = 16 байт)
- AES-192-CTR-LE (ключ = 24 байта, IV = 16 байт)
- AES-128-CTR-LE (ключ = 16 байт, IV = 16 байт)
- AES-256-CTR-BE (ключ = 32 байта, IV = 16 байт)
- AES-192-CTR-BE (ключ = 24 байта, IV = 16 байт)
- AES-128-CTR-BE (ключ = 16 байт, IV = 16 байт)
- AES-256-CBC-PKCS7 (ключ = 32 байта, iv = 16 байт)
- AES-192-CBC-PKCS7 (ключ = 24 байта, iv = 16 байт)
- AES-128-CBC-PKCS7 (ключ = 16 байт, iv = 16 байт)
- AES-256-CBC-ANSIX923 (ключ = 32 байта, iv = 16 байт)
- AES-192-CBC-ANSIX923 (ключ = 24 байта, iv = 16 байт)
- AES-128-CBC-ANSIX923 (ключ = 16 байт, iv = 16 байт)
- AES-256-CBC-ISO7816 (ключ = 32 байта, iv = 16 байт)
- AES-192-CBC-ISO7816 (ключ = 24 байта, iv = 16 байт)
- AES-128-CBC-ISO7816 (ключ = 16 байт, iv = 16 байт)
- AES-256-CBC-ISO10126 (ключ = 32 байта, iv = 16 байт)
- AES-192-CBC-ISO10126 (ключ = 24 байта, iv = 16 байт)
- AES-128-CBC-ISO10126 (ключ = 16 байт, iv = 16 байт)
- CHACHA20-POLY1305 (ключ = 32 байта, IV = 12 байт)
- XCHACHA20-POLY1305 (ключ = 32 байта, IV = 24 байта)
- XSALSA20-POLY1305 (ключ = 32 байта, IV = 24 байта)

Спецификация функции

```
decrypt(ciphertext: <строка>, algorithm: <строка>, key: <строка>, iv: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`ciphertext`	строка	Строка, которую необходимо расшифровать. Она должна представлять собой сырые байты (не закодированные).		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`algorithm`	строка	Алгоритм, который необходимо использовать.		да
`key`	строка	Ключ для расшифровки. Он должен представлять собой сырые (не закодированные) байты. Его длина должна соответствовать указанному алгоритму.		да
`iv`	строка	Вектор инициализации (IV) для расшифровки. Он должен представлять собой сырые (не закодированные) байты IV. Его длина должна соответствовать указанному алгоритму. Для каждого сообщения следует генерировать новый IV. Вы можете использовать `random_bytes` для генерации криптографически безопасного случайного значения. Значение должно совпадать с тем, что использовалось во время шифрования.		да

Ошибки

Функция `decrypt` **может возвращать ошибки**, для которых требуется обработка:

- Алгоритм `algorithm` не поддерживается.
- Длина `key` не соответствует требуемому размеру ключа для указанного алгоритма.
- Длина `iv` не соответствует требуемому размеру IV для указанного алгоритма.

Примеры

Расшифровка значения

Пример 1104. Исходный код

```
ciphertext = decode_base64!("5fLGcu1VHdzsPcGNDio7asLqE1P43QrVfPfmP4i4zOU=")
iv = decode_base64!("fVEIRkIiczCRWNxaarsyxA==")
key = "16_byte_keyxxxxx"
decrypt!(ciphertext, "AES-128-CBC-PKCS7", key, iv: iv)
```

Пример 1105. Результат

```
super_secret_message
```

Функции генерации случайных значений и ID

В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

Функции генерации случайных значений

random_bool

Возвращает случайное логическое значение.

Спецификация функции

```
random_bool()
:: <логическое значение>
```

Примеры

Случайное логическое значение

Пример 1106. Исходный код

```
is_boolean(random_bool())
```

Пример 1107. Результат

```
true
```

random_int

Возвращает случайное целое число в диапазоне [`min`, `max`).

Для более удобного синтаксиса используйте функцию `rv_random_int`.

Спецификация функции

```
random_int(min: <целое число>, max: <целое число>)
:: <целое число>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`min`	целое число	Минимальное значение (включительно)		да
`max`	целое число	Максимальное значение (исключительно)		да

Ошибки

Функция `random_int` **может возвращать ошибки**, для которых требуется обработка:

- `max` не больше `min`.

Примеры

Случайное целое число от 0 до 10 (не включая 10)

Пример 1108. Исходный код

```
i = random_int(0, 10)
i >= 0 && i < 10
```

Пример 1109. Результат

```
true
```

rv_random_int

Возвращает псевдослучайное целое число в диапазоне [`min`, `max`]. Любую из границ диапазона можно опустить, по умолчанию в качестве минимума используется `0`, а в качестве максимума — максимально возможное 64-битное число.

Спецификация функции

```
rv_random_int([min: <целое число>, max: <целое число>])
:: <целое число>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`min`	целое число	Минимальное значение (включительно)	`0`	нет
`max`	целое число	Максимальное значение (включительно)	$2^{63} - 1$	нет

Примеры

Случайное целое число от 0 до 10 (включая 10)

Пример 1110. Исходный код

```
i = rv_random_int(0, 10)
```

```
i >= 0 && i <= 10
```

Пример 1111. Результат

```
true
```

random_float

Возвращает случайное число с плавающей точкой в диапазоне [min, max).

Спецификация функции

```
random_float(min: <число с плавающей точкой>, max: <число с плавающей точкой>)
:: <число с плавающей точкой>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`min`	число с плавающей точкой	Минимальное значение (включительно).		да
`max`	число с плавающей точкой	Максимальное значение (исключительно).		да

Ошибки

Функция `random_float` **может возвращать ошибки**, для которых требуется обработка:

- `max` не больше `min`

Примеры

Случайное число с плавающей точкой от 0.0 до 10.0 (не включая 10.0)

Пример 1112. Исходный код

```
f = random_float(0.0, 10.0)
f >= 0 && f < 10
```

Пример 1113. Результат

```
true
```

random_bytes

Криптографически стойкий генератор случайных чисел. Возвращает строковое значение, содержащее запрошенное количество случайных байтов.

Спецификация функции

```
random_bytes(length: <целое число>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`length`	целое число	Количество генерируемых байт. Должно быть не больше 64 Кбайт.		да

Ошибки

Функция `random_bytes` **может возвращать ошибки**, для которых требуется обработка:

- `length` отрицательная
- `length` больше максимального значения (64 Кбайт)

Примеры

Генерация случайных байтов в кодировке base64

Пример 1114. Исходный код

```
encode_base64(random_bytes(16))
```

Пример 1115. Результат

```
LNu0BBgUbh7XAlXbjSOomQ==
```

Функции генерации ID

uuid_from_friendly_id

Конвертирует Friendly ID (base62-кодировка 128-битного слова) в UUID.

Спецификация функции

```
uuid_from_friendly_id(value: <метка времени> )
:: <строка> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	метка времени	Строка, являющаяся Friendly ID		да

Ошибки

Функция `uuid_from_friendly_id` **может возвращать ошибки**, для которых требуется обработка:

- **value** является строкой, но в тексте используются символы, не принадлежащие классу `[0-9A-Za-z]`.
- **value** является base62-кодировкой целого числа, но число больше либо равно 2^{128}.

Примеры

Конвертация Friendly ID в UUID

Пример 1116. Исходный код

```
uuid_from_friendly_id!("3s87yEvnmkiPBMHsj8bwwc")
```

Пример 1117. Результат

```
7f41deed-d5e2-8b5e-7a13-ab4ff93cfad2
```

uuid_v4

Генерирует случайную строку UUIDv4.

Спецификация функции

```
uuid_v4()
:: <строка>
```

Примеры

Создание UUIDv4

Пример 1118. Исходный код

```
uuid_v4()
```

Пример 1119. Результат

```
1d262f4f-199b-458d-879f-05fd0a5f0683
```

uuid_v7

Генерирует случайную строку UUIDv7. По умолчанию в качестве метки времени используется результат функции `now()`.

Спецификация функции

```
uuid_v7([timestamp: <метка времени>])
```

```
:: <строка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`timestamp`	метка времени	Метка времени, используемая для генерации UUIDv7.	`now()`	нет

Ошибки

Функция `to_unix_timestamp` **может возвращать ошибки**, для которых требуется обработка:

- `timestamp` вне допустимого диапазона для меток времени.

Примеры

Создание UUIDv7 с меткой времени по умолчанию

Пример 1120. Исходный код

```
uuid_v7()
```

Пример 1121. Результат

```
06338364-8305-7b74-8000-de4963503139
```

Создание UUIDv7 с явно указанной `now`

Пример 1122. Исходный код

```
uuid_v7(now())
```

Пример 1123. Результат

```
06338364-8305-7b74-8000-de4963503139
```

Создание UUIDv7 с произвольной меткой времени

Пример 1124. Исходный код

```
uuid_v7(t'2020-12-30T22:20:53.824727Z')
```

Пример 1125. Результат

```
0176b5bd-5d19-7394-bb60-c21028c6152b
```

community_id

Генерирует ID потока данных на основе спецификации Community ID.

Спецификация функции

```
community_id(source_ip: <строка>, destination_ip: <строка>, protocol: <целое число>,
[source_port: <целое число>, destination_port: <целое число>, seed: <целое число> ])
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`source_ip`	строка	IP-адрес источника.		да
`destination_ip`	строка	IP-адрес назначения.		да
`protocol`	целое число	Номер протокола.		да
`source_port`	целое число	Исходный порт или тип ICMP.		нет
`destination_port`	целое число	Порт назначения или код ICMP.		нет
`seed`	целое число	Произвольное входное число (seed) между 0 и 65 535.	`0`	нет

Ошибки

Функция `community_id` **может возвращать ошибки**, для которых требуется обработка:

- Невозможно разобрать IP-адрес источника.
- Невозможно разобрать IP-адрес назначения.
- Номер протокола вне диапазона [0; 255].
- Номер порта источника вне диапазона [0; 65535].
- Номер порта назначения вне диапазона [0; 65535].
- Указан `seed` вне диапазона [0; 65535].

Примеры

TCP

Пример 1126. Исходный код

```
community_id!(source_ip: "192.0.2.0",
destination_ip: "198.51.100.0",
source_port: 1122,
destination_port: 3344,
protocol: 6)
```

Пример 1127. Результат

```
1:yQ6Snpz/rq5kekU4Yqbt5zCjHWM=
```

UDP

Пример 1128. Исходный код

```
community_id!(source_ip: "192.0.2.0",
destination_ip: "198.51.100.0",
source_port: 1122,
destination_port: 3344,
protocol: 17)
```

Пример 1129. Результат

```
1:o1hyPRzWSLSK8hSMFSCDlkppQQ0=
```

ICMP

Пример 1130. Исходный код

```
community_id!(source_ip: "192.0.2.0",
destination_ip: "198.51.100.0",
source_port: 1122,
destination_port: 3344,
protocol: 1)
```

Пример 1131. Результат

```
1:mrrlg1BzCI7PkjFBx+q1431HpLg=
```

RSVP

Пример 1132. Исходный код

```
community_id!(source_ip: "192.0.2.0",
destination_ip: "198.51.100.0",
source_port: 1122,
destination_port: 3344,
protocol: 46)
```

Пример 1133. Результат

```
1:fktxP6ndCPNrcDJ2XpM5DFoWtT4=
```

Системные функции

В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

get_env_var

Возвращает значение переменной окружения, указанной в `name`.

Спецификация функции

```
get_env_var(name: <строка>)
:: <строка>, <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`name`	строка	Имя переменной окружения		да

Ошибки

Функция `get_env_var` **может возвращать ошибки**, что означает, что обработка необходима для следующих ошибок:

- Переменная окружения `name` не существует.
- Значение переменной окружения `name` не является допустимым значением Unicode.

Примеры

Получить значение переменной окружения

Пример 1134. Исходный код

```
get_env_var!("HOME")
```

Пример 1135. Результат

```
/root
```

get_hostname

Возвращает имя хоста локальной системы.

Спецификация функции

```
get_hostname()
:: <строка>, <ошибка>
```

Ошибки

Функция `get_hostname` **может возвращать ошибки**, для которых требуется обработка:

- Внутреннее разрешение имени хоста завершилось неудачей.

Примеры

Получить имя хоста

Пример 1136. Исходный код

```
.hostname = get_hostname!()
```

get_timezone_name

Возвращает название часового пояса, заданного в конфигурации Vector. Если в конфигурации установлено значение local, то функция пытается определить часовой пояс на основе операционной системы хоста. Если определить название согласно IANA не получилось, возвращается смещение местного часового пояса от UTC в формате `"[-+] ЧЧ:ММ"`, например, `"+02:00"`.

Спецификация функции

```
get_timezone_name()
:: <строка>, <ошибка>
```

Ошибки

Функция `get_timezone_name()` **может возвращать ошибки**, для которых требуется обработка:

- Получение информации о местном часовом поясе завершилось неудачей.

Пример

Получить имя местного часового пояса согласно IANA

Пример 1137. Исходный код

```
.vector_timezone = get_timezone_name!()
```

now

Возвращает текущую отметку времени в часовом поясе UTC с точностью до наносекунд.

Спецификация функции

```
now()
:: <метка времени>
```

Примеры

Получение текущей отметки времени

Пример 1138. Исходный код

```
now()
```

Пример 1139. Результат

```
2021-03-04T10:51:15.928937Z
```

Функции отладки

В данном разделе при описании функций приняты следующие обозначения:

- Типы значений, которые принимает аргумент, указаны в угловых скобках.
- Перечень необязательных аргументов заключается в квадратные скобки. При вызове функции квадратные скобки вводить не нужно.
- После `::` в угловых скобках указаны типы значений, которые возвращает функция.

assert

Проверяет условие `condition`, которое должно возвращать логическое значение. Если условие оценивается как `false`, программа завершается с сообщением `message`.

Спецификация функции

```
assert(condition: <логическое значение | сравнение | логическое выражение>, [message:
<строка>])
:: <пустое значение> , <ошибка>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`condition`	логическое значение сравнение логическое выражение	Условие для проверки, должно возвращать логическое значение.		да

Аргумент	Тип	Описание	По умолчанию	Обязателен
`message`	строка	Необязательное пользовательское сообщение об ошибке. Если утверждение условия не выполняется, к пользовательскому сообщению автоматически добавляется префикс сообщения по умолчанию. См. примеры ниже для примера полностью сформированного сообщения в журнале.		нет

Примечания

Эта функция имеет специальное поведение, о котором следует знать.

- Функцию `assert` следует использовать автономно и только тогда, когда вы хотите прервать программу. Избегайте ее использования в логических выражениях и других ситуациях, в которых вы хотите, чтобы программа продолжала работу, даже если условие оценивается как `false`.

Ошибки

Функция `assert` **может возвращать ошибки**, для которых требуется обработка:

- `условие` оценивается как `false`

Примеры

Утверждение (true)

Пример 1140. Исходный код

```
ok, err = assert("foo" == "foo", message: "\"foo\" must be \"foo\"!")
```

Пример 1141. Результат

```
true
```

Утверждение (false)

Пример 1142. Исходный код

```
assert!("foo" == "bar", message: "\"foo\" must be \"foo\"!")
```

assert_eq

Проверяет, что два выражения, `left` и `right`, имеют одинаковое значение. Если они не равны, программа прерывается с сообщением `message`.

Спецификация функции

```
assert_eq(left: <любой тип>, right: <любой тип>, [message: <строка>])
:: <логическое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`left`	любое	Значение для проверки на равенство с `right`.		да
`right`	любое	Значение для проверки на равенство с `left`.		да
`message`	строка	Необязательное пользовательское сообщение об ошибке. Если проверка на равенство не выполняется, к пользовательскому сообщению автоматически добавляется префикс сообщения по умолчанию. См. примеры ниже для примера полностью сформированного сообщения в журнале.		нет

Примечания

Эта функция имеет специальное поведение, о котором следует знать.

- Функцию `assert_eq` следует использовать автономно и только тогда, когда вы хотите прервать программу. Избегайте ее использования в логических выражениях и других ситуациях, в которых вы хотите, чтобы программа продолжала работу, даже если условие оценивается как `false`.

Примеры

Успешное утверждение

Пример 1143. Исходный код

```
assert_eq!(1, 1)
```

Пример 1144. Результат

```
true
```

Неудачное утверждение

Пример 1145. Исходный код

```
assert_eq!(127, [1, 2, 3])
```

Неудачное утверждение с пользовательским сообщением в журнале

Пример 1146. Исходный код

```
assert_eq!(1, 0, message: "Unequal integers")
```

log

Выводит `value` в стандартный вывод Vector с указанным `level`.

Спецификация функции

```
log(value: <любой тип>, [level: <строка>, rate_limit_secs: <целое число>])
:: <пустое значение>
```

Аргумент	Тип	Описание	По умолчанию	Обязателен
`value`	любое	Значение для вывода в лог.		да
`level`	строка	Уровень логирования.	`info`	нет
`rate_limit_secs`	целое число	Определяет, что сообщение будет выводиться не чаще указанного количества секунд. Используйте значение `0`, чтобы отключить ограничение частоты.	`1`	нет

Примеры

Вывод сообщения в лог

Пример 1147. Исходный код

```
log("Hello, World!", level: "info", rate_limit_secs: 60)
```

Пример 1148. Результат

```
null
```

Вывод ошибки в лог

Пример 1149. Исходный код

```
ts, err = to_int(.field)
if err != null {
  log(err, level: "error")
}
```

Пример 1150. Результат

```
null
```

Справочник ошибок

100 Unhandled root runtime error

Корневое выражение может вызывать ошибку времени выполнения, и она не обрабатывается в программе VRL.

Обоснование

VRL надежно обрабатывает ошибки и требует, чтобы все возможные ошибки времени выполнения были обработаны. Это гарантирует безотказную работу VRL и помогает обеспечить надежное выполнение программ VRL при их развертывании.

Решение

Обработайте ошибку времени выполнения путем присвоения, объединения или вызова ошибки.

Примеры

Необработанная ошибка времени выполнения корневого выражения (присвоение)

Пример 1151. Программа VRL

```
get_env_var("HOST")
```

Пример 1152. Как исправить

```
.host = get_env_var("HOST")
```

101 Malformed regex literal

Выражение литерала регулярного выражения сформировано неправильно и, следовательно, не приводит к действительному регулярному выражению.

Обоснование

Недопустимые регулярные выражения не компилируются.

Решение

Регулярные выражения часто приводят к ошибкам синтаксиса. Если вы обрабатываете общий формат лога (Common Log Format), рекомендуется использовать одну из функций `parse_*` VRL.

Если вы не обнаружили подходящей функции для вашего формата, попробуйте использовать инструмент тестирования регулярных выражений Rust для тестирования и исправления вашего регулярного выражения.

Примеры

Неверный литерал регулярного выражения (общий формат)

Пример 1153. Программа VRL

```
. |= parse_regex!(.message, r'^(?P<host>[\w\.]+) - (?P<user>[\w]+) (?P<bytes_in>[\d]+)
\[?P<timestamp>.*)\] "(?P<method>[\w]+) (?P<path>.*)" (?P<status>[\d]+)
(?P<bytes_out>[\d]+)$')
```

Пример 1154. Как исправить

```
. |= parse_common_log!(.message)
```

102 Non-boolean if expression predicate

Предикат условного выражения не вычисляется в значение логического типа.

Обоснование

VRL не поддерживает "истинные" значения (нелогические значения, которые приводятся к значению `true`, например, `1`), так как они могут привести к неожиданному поведению при использовании в условных выражениях. Это гарантирует безопасную работу VRL и надежное выполнение программ VRL при их развертывании.

Решение

Измените предикат условного выражения так, чтобы он вычислялся в значение логического типа. Полезной функцией для решения этой проблемы является `is_nullish`.

Примеры

Предикат условного выражения с логическим типом (строки)

Пример 1155. Событие журнала

```
{
  "message": "key=value"
}
```

Пример 1156. Программа VRL

```
if .message {
. |= parse_key_value!(.message)
}
```

Пример 1157. Как исправить

```
if exists(.message) {
. |= parse_key_value!(.message)
}
```

103 Unhandled fallible assignment

Правая часть этого присваивания может вызывать ошибку времени выполнения, но ошибка не обрабатывается.

Обоснование

VRL надежно обрабатывает ошибки и требует, чтобы все возможные ошибки времени выполнения были обработаны. Это гарантирует безотказную работу VRL и помогает обеспечить надежное выполнение программ VRL при их развертывании.

Решение

Обработайте ошибку времени выполнения путем присвоения, объединения или вызова ошибки.

Примеры

Необработанное присваивание с возвратом ошибки (объединение)

Пример 1158. Событие журнала

```
{
  "message": "key=value"
}
```

Пример 1159. Программа VRL

```
. = parse_key_value(.message)
```

Пример 1160. Как исправить

```
. = parse_key_value(.message) ?? {}
```

Необработанное присваивание с возвратом ошибки (вызов ошибки)

Пример 1161. Событие журнала

```
{
  "message": "key=value"
}
```

Пример 1162. Программа VRL

```
. = parse_key_value(.message)
```

Пример 1163. Как исправить

```
. = parse_key_value!(.message)
```

Необработанное присваивание с возвратом ошибки (присвоение)

Пример 1164. Событие журнала

```
{
  "message": "key=value"
}
```

Пример 1165. Программа VRL

```
. = parse_key_value(.message)
```

Пример 1166. Как исправить

```
., err = parse_key_value(.message)
```

Необработанное присваивание после изменения типа переменной внутри замыкания

Пример 1167. Программа VRL

```
parsed_value = {}
for_each(object!(.value)) -> |key, value| {
    kv = parse_json!(value)
    parsed_value = set!(parsed_value, [key], kv)
}
result = ""
for_each(object!(parsed_value)) -> |_asd, value| {
  result = result + value.id + value.level + " "
}
```

Пример 1168. Как исправить

```
result = result + value.id + value.level + " " ?? "fallback"
```

104 Unnecessary error assignment

Левая часть выражения присваивания обрабатывает ошибки, хотя правая часть *не может* вызвать ошибку.

Обоснование

Присваивание ошибки, когда это невозможно, фактически является мертвым кодом, который затрудняет чтение программы. Удаление присваивания ошибки упростит вашу программу.

Решение

Удалите присваивание ошибки.

Примеры

Ненужное присваивание ошибки (строки)

Пример 1169. Программа VRL

```
.message, err = downcase(.message)
```

Пример 1170. Как исправить

```
.message = downcase(.message)
```

105 Undefined function

Вызывается неизвестная функция.

Решение

Ошибка обычна связана с опечатками. Исправление имени функции должно решить эту проблему.

Примеры

Неопределенная функция (опечатка)

Пример 1171. Программа VRL

```
parse_keyvalue(.message)
```

Пример 1172. Как исправить

```
parse_key_value(.message)
```

106 Function argument arity mismatch

Выражение вызова функции вызывает функцию слишком большим перечнем аргументов.

Решение

Удалите лишние аргументы, чтобы соответствовать задокументированной сигнатуре функции.

Примеры

Несоответствие количества аргументов функции

Пример 1173. Программа VRL

```
parse_json(.message, pretty: true)
```

Пример 1174. Как исправить

```
parse_json(.message)
```

107 Required function argument missing

Выражение вызова функции не передает обязательный аргумент.

Решение

Укажите все обязательные аргументы функции, чтобы соответствовать задокументированной сигнатуре функции.

Примеры

Отсутствует обязательный аргумент функции

Пример 1175. Программа VRL

```
parse_timestamp(.timestamp)
```

Пример 1176. Как исправить

```
parse_timestamp(.timestamp, format: "%D")
```

108 Unknown function argument keyword

Выражение вызова функции содержит неизвестное именованное значение аргумента.

Решение

Исправьте имя так, чтобы оно соответствовало задокументированным именам аргументов функции.

Примеры

Неизвестное ключевое слово аргумента функции

Пример 1177. Программа VRL

```
parse_timestamp(.timestamp, fmt: "%D")
```

Пример 1178. Как исправить

```
parse_timestamp(.timestamp, format: "%D")
```

110 Invalid argument type

Аргумент, переданный выражению функции, не является поддерживаемым типом.

Обоснование

VRL надежно обрабатывает ошибки и требует, чтобы все возможные ошибки времени выполнения были обработаны. Это гарантирует безотказную работу VRL и помогает обеспечить надежное выполнение программ VRL при их развертывании.

Решение

Обеспечьте тип переменной, используя соответствующий тип или функцию приведения типов.

Примеры

Неверный тип аргумента (защита с использованием значений по умолчанию)

Пример 1179. Программа VRL

```
downcase(.message)
```

Пример 1180. Как исправить

```
.message = string(.message) ?? ""
downcase(.message)
```

Неверный тип аргумента (защита с использованием ошибок)

Пример 1181. Программа VRL

```
downcase(.message)
```

Пример 1182. Как исправить

```
downcase(string!(.message))
```

111 Unhandled predicate error

Предикат вызывает ошибку времени выполнения, и она не обрабатывается в программе VRL.

Обоснование

VRL надежно обрабатывает ошибки и требует, чтобы все возможные ошибки времени выполнения были обработаны. Это гарантирует безотказную работу VRL и помогает обеспечить надежное выполнение программ VRL при их развертывании.

Решение

Обработайте ошибку времени выполнения путем присвоения, объединения или вызова ошибки.

Примеры

Необработанная ошибка предиката (предикат)

Пример 1183. Программа VRL

```
if contains(.field, "thing") {
  log("thing")
```

```
}
```

Пример 1184. Как исправить

```
if contains(.field, "thing") ?? false {
  log("thing")
}
```

203 Unrecognized token

Ваша программа VRL содержит токен (символ), который парсер VRL не распознает как допустимый.

Решение

Используйте допустимый токен.

Примеры

Нераспознанный токен

Пример 1185. Программа VRL

```
😁
```

Пример 1186. Как исправить

```
#некоторое допустимое значение
"😁"
```

204 Unrecognized end-of-file (EOF)

Парсер VRL достиг конца программы в недопустимом состоянии, возможно, из-за опечатки или "висячего" выражения.

Решение

Убедитесь, что последнее выражение в программе является допустимым.

Примеры

Нераспознанный конец файла (EOF)

Пример 1187. Программа VRL

```
.field1 = "value1"
.field2 =
```

Пример 1188. Как исправить

```
.field1 = "value1"
.field2 = "value2"
```

205 Reserved keyword

Вы использовали имя переменной, которое служит для другой цели в VRL или зарезервировано для возможного будущего использования.

Решение

Используйте другое имя переменной.

Примеры

Зарезервированное ключевое слово

Пример 1189. Программа VRL

```
else = "некое значение"
```

Пример 1190. Как исправить

```
some_non_reserved_name = "некое значение"
```

206 Invalid numeric literal

Компилятор VRL не распознает числовой литерал как допустимый.

207 Invalid string literal

Программа VRL содержит строковый литерал, который парсер VRL не распознает как допустимый.

Решение

Убедитесь, что строка правильно заключена в одинарные или двойные кавычки.

Примеры

Недопустимый строковый литерал

Пример 1191. Программа VRL

```
"Houston, we have a problem'
```

Пример 1192. Как исправить это

```
"Houston, we have a problem"
```

208 Invalid literal

Компилятор VRL не распознает литерал как допустимое значение.

209 Invalid escape character

Строка содержит символ экранирования, который компилятор VRL не распознает как допустимый.

300 Unexpected type

Компилятор VRL ожидал значение определенного типа, но обнаружил другой тип.

301 Type coercion error

Значение не может быть приведено к желаемому типу.

302 Remainder error

Типы не могут дать остаток.

303 Multiplication error

Типы не могут быть перемножены.

304 Division error

Левое значение не может быть разделено на правое значение.

305 Divide by zero

Попытка разделить целое число или число с плавающей точкой на ноль.

Обоснование

В отличие от некоторых других языков программирования, VRL не имеет понятия бесконечности, так как неясно, как это может быть применимо к использованию данных для наблюдения. Таким образом, деление на ноль не может иметь осмысленного результата.

Решение

Если вы знаете, что значение **обязательно** равно нулю, не делите на него. Если значение **может быть** нулем, обработайте потенциальную ошибку, вызванную операцией:

```
result, err = 27 / .some_value
if err != nil {
# Обработать ошибку
}
```

306 NaN float

Числа с плавающей точкой в VRL не могут иметь значение NaN (не число).

307 Addition error

Значения не могут быть сложены.

Для соединения массивов используйте функцию append.

308 Subtraction error

Правое значение не может быть вычтено из левого значения.

309 Or expression error

Значения не могут быть объединены в выражение OR.

310 And expression error

Значения не могут быть объединены в выражение AND.

311 Greater than error

Значения нельзя использовать в выражении "больше чем".

312 Greater than or equal to error

Значения нельзя использовать в выражении "больше или равно".

313 Less than error

Эти два значения нельзя использовать в выражении "меньше чем".

314 Less than or equal to error

Эти два значения нельзя использовать в выражении "меньше или равно".

315 mutation of read-only value

Это значение доступно только для чтения и не может быть удалено или изменено.

400 Unexpected expression

Компилятор VRL встретил тип выражения, которое здесь не ожидалось.

401 Invalid enum variant

VRL ожидает значение перечисления для этого аргумента, но значение, которое вы ввели для перечисления, недопустимо.

Решение

Проверьте документацию для этой функции в справочнике функций VRL, чтобы узнать, какие значения перечисления допустимы для этого аргумента.

402 Expected static expression for function argument

VRL ожидал статическое выражение для аргумента функции, но было предоставлено динамическое выражение (например, переменная).

Для некоторых аргументов функций VRL требуются статические выражения для проверки типов аргументов на этапе компиляции, чтобы избежать ошибок времени выполнения.

Решение

Замените динамический аргумент статическим выражением.

403 Invalid argument

Функции передан недопустимый аргумент. Строка ошибки содержит более подробные сведения о том, что было недопустимо.

Решение

Проверьте строку этой ошибки, чтобы узнать, что было недопустимо.

601 Invalid timestamp

Предоставленный литерал метки времени корректно оформлен (т. е. использует синтаксис `t'…'`), но метка времени не соответствует формату RFC 3339.

Обоснование

Недопустимые метки времени не компилируются.

Решение

Приведите метку времени в соответствие с форматом RFC 3339.

Примеры

Неверное форматирование метки времени

Пример 1193. Программа VRL

```
.timestamp = format_timestamp!(t'next Tuesday', format: "%v %R")
```

Пример 1194. Как исправить

```
.timestamp = format_timestamp!(t'2021-03-09T16:33:02.405806Z', format: "%v %R")
```

620 Aborting infallible function

Вы указали, что функция должна прерываться при ошибке, хотя функция не возвращает ошибки.

Обоснование

В VRL функции, которые не возвращают ошибки — это функции, которые не могут завершиться с ошибкой и не требуют обработки ошибок. Это означает, что не имеет смысла использовать `!` в вызове функции для прерывания при ошибке.

Решение

Удалите `!` из вызова функции.

Примеры

Прерывание невосприимчивой функции

Пример 1195. Программа VRL

```
encode_json!(["one", "two", "three"])
```

Пример 1196. Как исправить

```
encode_json(["one", "two", "three"])
```

630 Fallible argument

Вы передали выражение, которое возвращает ошибки, в качестве аргумента функции.

Обоснование

В VRL выражения, которые вы передаете функциям в качестве аргументов, сами также не должны возвращать ошибки. В противном случае результат функции будет неопределенным.

Решение

Сделайте так, чтобы выражение, передаваемое в функцию, не могло возвращать ошибки, потенциально прервав его при возникновении ошибки с помощью `!`, объединив его ошибку с помощью `??` или с помощью некоторого другого метода.

Примеры

Аргумент, возвращающий ошибки

Пример 1197. Программа VRL

```
format_timestamp!(parse_timestamp!("2021-01-17T23:27:31.891948Z", "%+"), format: "%v
%R")
```

Пример 1198. Как исправить

```
format_timestamp!(parse_timestamp!("2021-01-17T23:27:31.891948Z", "%+"), format: "%v
%R")
```

631 Fallible abort message expression

Вы передали выражение, которое возвращает ошибки, в качестве сообщения для прерывания.

Обоснование

Выражение, которое вы передаете для прерывания, не должно возвращать ошибки. В противном случае выражение прерывания может завершиться с ошибкой во время выполнения.

Решение

Сделайте так, чтобы выражение, передаваемое в функцию, не могло возвращать ошибки, потенциально прервав его при возникновении ошибки с помощью !, объединив его ошибку с помощью ?? или с помощью некоторого другого метода.

Примеры

Выражение с сообщением о прерывании (может возвращать ошибку)

Пример 1199. Программа VRL

```
abort to_syslog_level(0)
```

Пример 1200. Как исправить

```
abort to_syslog_level(0) ?? "other"
```

640 No-op assignment

Вы присвоили значение чему-то, что не является ни переменной, ни путем.

Обоснование

Все присваивания в VRL должны быть либо путем, либо переменной. Если вы пытаетесь присвоить значение, например, подчеркиванию (_), эта операция считается "бездействующей" (no-op), так как она не имеет эффекта (и, следовательно, не является присваиванием).

Решение

Присвойте значение, содержащееся в правой части выражения, переменной или пути.

Примеры

Присваивание значения

Пример 1201. Программа VRL

```
_ = "пример строки"
```

Пример 1202. Как исправить

```
# .some_string = "пример строки"
```

650 Chained comparison operators

Вы связали несколько операторов сравнения вместе таким образом, что не может получиться допустимое выражение.

Обоснование

Операторы сравнения могут работать только с двумя операндами, например, `1 != 2`. Связывание их вместе, например, `1 < 2 < 3`, приводит к бессмысленному неприменимому выражению.

Решение

Используйте операторы сравнения только с левым и правым значением. Вы **можете** связывать операции сравнения вместе при условии, что выражения правильно сгруппированы. Так, например, `a == b == c` не является допустимым, но `a == b && b == c` **является** допустимым, так как в нем используются отдельные логические выражения.

Примеры

Цепочка операторов сравнения

Пример 1203. Программа VRL

```
1 == 1 == 2
```

Пример 1204. Как исправить

```
(1 == 1) && (1 == 2)
```

651 Unnecessary error coalescing operation

Вы использовали операцию объединения (`??`) для обработки ошибки, но в данном случае левая операция не может возвращать ошибки, и поэтому правое значение после `??` никогда не достигается.

Обоснование

Операции объединения полезны, когда вы хотите указать, что должно произойти, если операция завершается с ошибкой. Например:

```
result = op1 ?? op2
```

В этом примере, если `op1` не возвращает ошибки (не может завершиться с ошибкой), то переменная `result` устанавливается в значение `op1`, а `op2` никогда не достигается.

Решение

Если левая операция не должна возвращать ошибки, удалите оператор `??` и правую операцию. Однако, если левая операция должна возвращать ошибки, удалите `!` из вызова функции и все остальное, что приводит к вызову ошибок.

652 Only objects can be merged

Вы пытаетесь слить два значения, но одно или оба из них не являются объектами.

Обоснование

Среди типов, доступных в VRL, только объекты могут быть слиты. Неясно, что означало бы слияние, например, объекта с логическим значением. Однако следует отметить, что для некоторых других типов VRL доступны операции, похожие на слияние:

- Строки можно складывать.
- Массивы можно присоединять к другим массивам функцией append.

Эти операции могут оказаться полезными, если вы случайно использовали `merge`.

Решение

Убедитесь, что оба значения, которые вы объединяете, являются объектами VRL. Если вы не уверены, является ли значение объектом, вы можете использовать функцию `object` для проверки.

660 Non-Boolean negation

Вы использовали оператор отрицания для отрицания нелогического выражения.

Обоснование

Оператор отрицания (`!`) может использоваться только с логическими значениями. Например, выражение `!false` приводит к значению `true`, в то время как `!"hello"` является бессмысленным выражением.

Решение

Используйте оператор отрицания только с логическими выражениями.

Примеры

Отрицание нелогических значений

Пример 1205. Программа VRL

```
!47
```

Пример 1206. Как исправить

```
!(47 == null)
```

701 Call to Undefined Variable

Ссылка на неопределенную переменную.

Обоснование

Ссылка на переменную, которая не определена, приводит к неожиданному поведению и, скорее всего, связана с опечаткой.

Решение

Сначала присвойте переменной значение или исправьте опечатку в ссылке.

Примеры

Неопределенная переменная

Пример 1207. Программа VRL

```
my_variable
```

Пример 1208. Как исправить

```
my_variable = true
my_variable
```

Неправильное имя переменной

Пример 1209. Программа VRL

```
my_variable = true
my_var
```

Пример 1210. Как исправить

```
my_variable = true
my_variable
```

801 Usage of deprecated item

Ссылка на устаревший элемент. Обычно предлагается альтернатива, которую можно использовать вместо нее.

Обоснование

Этот элемент будет удален в будущем.

Решение

Примените предложенную альтернативу.

Советы по написанию кода на языке VRL

В данном разделе представлены рекомендации по написанию качественного кода на языке VRL.

Содержание раздела:

- Оптимизация кода
- Оформление кода

Оптимизация кода

В данном разделе рассматриваются методы оптимизации кода VRL, направленные на повышение производительности системы и упрощение работы с кодом.

Оптимизация VRL-кода включает минимизацию избыточных операций, таких как лишнее копирование объектов, неоптимальные вычисления и использование устаревших или неэффективных функций.

Снижение вычислительных затрат

Минимизация копирования исходного объекта

Копирование исходного объекта в VRL-коде может привести к избыточным затратам ресурсов, особенно если оно выполняется для каждого события. Для повышения производительности такие операции следует минимизировать.

При написании логики обработки может встречаться лишнее копирование объекта, как в данном примере:

Пример 1211. Код с избыточным копированием

```
...
on_correlate: !vrl |
  state = .
  %state.check = false
  for_each(["управляемые устройства/задача обновления"]) -> |_index, value| {
    if contains(downcase(to_string!(.msg) ?? ""), value) {
      %state.check = true
```

```
      %state.cs8 = .msg
      %state.cs8Label = "Antivirus Message"
      %state.msg = "На хосте {{ to_string(.dvc) ?? '-' }} было зафиксировано изменение
параметров критичной политики пользователем {{ to_string(.duser) ?? '-' }}"
    }
  }
  %state = state
...
```

Для устранения проблемы рекомендуется минимизировать операции копирования, сразу работая с переменной `%state`:

Пример 1212. Оптимизированный код

```
...
on_correlate: !vrl |
  %state = .
  %state.check = false
  for_each(["управляемые устройства/задача обновления"]) -> |_index, value| {
    if contains(downcase(to_string!(.msg) ?? ""), value) {
      %state.check = true
      %state.cs8 = .msg
      %state.cs8Label = "Antivirus Message"
      %state.msg = "На хосте {{ to_string(.dvc) ?? '-' }} было зафиксировано изменение
параметров критичной политики пользователем {{ to_string(.duser) ?? '-' }}"
    }
  }
...
```

Предотвращение дублирования логики

Другой пример избыточного копирования можно найти в следующем коде, где объект `state` дублируется на каждом событии:

Пример 1213. Код с дублирующимся копированием

```
...
on_correlate: !vrl |
  state = .
  count_detections = int(%state.count_detections) ?? 0
  count_detections = count_detections + 1
  %state.count_detections = count_detections
  if %state.count_detections == 2 {
    %state.cs8 = .msg
    %state.cs8Label = "Antivirus Message"
    %state.msg = "На хосте {{ to_string(.dvc) ?? '-' }} у пользователя {{
to_string(.duser) ?? '-' }} было обнаружено повторное заражение ВПО"
  }
  %state = state
```

```
...
```

Код можно оптимизировать, убрав лишнее копирование объекта `state`, что предотвратит ненужные срабатывания логики:

Пример 1214. Оптимизированный код

```
...
on_correlate: !vrl |
  if %state.check != true {
    %state.count_detections = (int(%state.count_detections) ?? 0) + 1
    if %state.count_detections == 2 {
      %state.check = true
      %state.cs8 = .msg
      %state.cs8Label = "Antivirus Message"
      %state.msg = "На хосте {{ to_string(.dvc) ?? '-' }} у пользователя {{
to_string(.duser) ?? '-' }} было обнаружено повторное заражение ВПО"
    }
  }
...
```

Оптимизация вычислений внутри циклов

При обработке событий в секции `on_correlate` все вычисления, не зависящие от цикла, следует выносить за его пределы. Это позволяет избежать повторных затрат ресурсов на каждой итерации цикла и повышает производительность. Например, выражение `downcase(to_string(.msg) ?? "-")` не изменяется в ходе выполнения цикла, но будет вычисляться на каждой итерации, если его не вынести в отдельную переменную.

Пример 1215. Пример с избыточными вычислениями

```
...
on_correlate: !vrl |
  # Список критичных объектов для мониторинга
  critical_objects = ["управляемые устройства/задача обновления"]
  %state = .
  %state.check = false
  for_each(critical_objects) -> |_index, value| {
    if contains(downcase(to_string(.msg) ?? "-"), value) {
      %state.check = true
      %state.cs8 = .msg
      %state.cs8Label = "Antivirus Message"
      %state.msg = "На хосте {{ to_string(.dvc) ?? '-' }} было зафиксировано изменение
параметров критичной политики пользователем {{ to_string(.duser) ?? '-' }}"
    }
  }
...
```

В данном варианте `downcase(to_string(.msg) ?? "-")` выполняется на каждой итерации, что

приводит к ненужным вычислительным затратам.

Пример 1216. Оптимизированный код

```
on_correlate: !vrl |
  # Список критичных объектов для мониторинга
  critical_objects = ["управляемые устройства/задача обновления"]
  # Оптимизация: предварительное вычисление значения
  downcase_msg = downcase(to_string(.msg) ?? "-")
  %state = .
  %state.check = false
  for_each(critical_objects) -> |_index, value| {
    if contains(downcase_msg, value) {
      %state.check = true
      %state.cs8 = .msg
      %state.cs8Label = "Antivirus Message"
      %state.msg = "На хосте {{ to_string(.dvc) ?? '-' }} было зафиксировано изменение
параметров критичной политики пользователем {{ to_string(.duser) ?? '-' }}"
    }
  }
```

В данном примере выражение `downcase(to_string(.msg) ?? "-")` вычисляется один раз до начала цикла и сохраняется в переменной `downcase_msg`. Это устраняет необходимость повторного выполнения вычисления на каждой итерации. В результате оптимизации количество вызовов функций сокращается, что снижает общий объем вычислений, особенно при большом количестве элементов в массиве `critical_objects`.

Исключение избыточных вычислений при повторяющихся проверках

Когда в логике обработки данных необходимо многократно выполнять одну и ту же проверку, это может привести к избыточным вычислениям и увеличению нагрузки на систему. В VRL отсутствует механизм для прерывания цикла, но избыточные вычисления можно исключить, используя функции, такие как `rv_contains_any` или `rv_ends_with_any`, которые выполняют поиск за один проход по строке.

Пример 1217. Код с использованием `rv_contains_any`

```
...
aliases:
  event:
    filter: !vrl |
      filePath = to_string!(.filePath) ?? ""
      rv_contains_any(filePath, [
        "/var/ossec/logs",
        "/var/run/utmp",
        "/var/log/messages",
        "/var/log/audit",
        "/var/log/secure",
        "/var/log/auth.log",
        "/var/log/kern.log",
        "/var/log/syslog",
```

```
        "/var/log/cron.log"
      ])
...
```

Пример 1218. Код с использованием `rv_ends_with_any`

```
...
aliases:
  event:
    filter: !vrl |
      filePath = to_string!(.filePath) ?? ""
      rv_ends_with_any(filePath, [
        ".bash_history",
        ".log",
        ".audit"
      ])
...
```

Для обработки условий на наличие подстрок или совпадений строк можно также использовать функции contains или match-any. Выбор функции зависит от количества условий и требований к производительности.

Функция `contains` работает эффективно при небольшом количестве условий, например, 2–3 условия.

Пример 1219. Код с использованием `contains`

```
...
aliases:
  event:
    filter: !vrl |
      filePath = to_string!(.filePath) ?? ""
      contains(filePath, "/var/log/audit") ||
      ends_with(filePath, ".bash_history")
...
```

При большом числе условий (5 и более) предпочтительнее использовать `match_any`, поскольку это упрощает код и снижает вероятность ошибок при добавлении новых условий.

Пример 1220. Код с использованием `match_any`

```
...
aliases:
  event:
    filter: !vrl |
      filePath = to_string!(.filePath) ?? ""
      match_any(filePath, [
        "/var/ossec/logs",
        "/var/run/utmp",
        "/var/log/messages",
        "/var/log/audit",
```

```
        "/var/log/secure",
        "/var/log/auth.log",
        "/var/log/kern.log",
        "/var/log/syslog",
        "/var/log/cron.log",
        ".bash_history"
      ])
...
```

Рекомендации по выбору функций

- `contains`: используйте для небольшого количества условий (2–3 условия), где нужно проверить наличие конкретных подстрок.
- `match_any`: предпочтительнее при большом количестве условий (5 и более условий), так как она упрощает код и повышает производительность.
- `rv_contains_any` или `rv_ends_with_any`: используйте для проверки наличия хотя бы одной подстроки или суффикса, так как эти функции выполняют поиск за один проход, обеспечивая лучшую производительность.

Учитывайте чувствительность к регистру, если это необходимо, и используйте дополнительный параметр `case_sensitive: false`.

Использование регулярных выражений

Регулярные выражения требуют больше ресурсов, и их стоит использовать только там, где не могут быть использованы другие функции VRL, например, для разбора конкретных форматов.

Пример 1221. Пример использования регулярных выражений

```
...
on_correlate: !vrl |
  message = parse_regex_all!(.raw, r'(?P<key_value>(\S*))')
...
```

Пример 1222. Оптимизированный код

```
...
on_correlate: !vrl |
  parsed_message = parse_key_value(.raw) ?? {}
...
```

Контроль структуры кода

Приоритеты операторов

При использовании нескольких логических операторов важно явно задавать приоритет операций с помощью скобок, чтобы избежать двусмысленных условий.

Если написать код без скобок:

```
.dvendor == "Microsoft" && .externalId == "4688" || .externalId == "4688"
```

Этот код будет интерпретирован как два отдельных блока:

- `.dvendor == "Microsoft"` И `.externalId == "4688"`
- `.externalId == "4688"`

Если цель заключается в том, чтобы условие `.dvendor == "Microsoft"` было связано с одним из значений `.externalId`, необходимо использовать скобки для группировки операторов.

Пример 1223. Пример с использованием скобок

```
.dvendor == "Microsoft" && (.externalId == "4688" || .externalId == "1234")
```

Для сложных условий с несколькими процессами также требуется группировка:

```
.dvendor == "Microsoft" && (.dproc == "ipconfig.exe" || .dproc == "nltest.exe" ||
.dproc == "ping.exe")
```

Приведение полей к нужному типу

Рекомендуется приводить значения полей к ожидаемому типу данных. Это помогает избежать ошибок при обработке данных и обеспечивает корректное выполнение кода.

Рекомендуемые значения для различных типов данных:

Таблица 41. Типы данных и их значения по умолчанию

Тип данных	Значение по умолчанию
`LCString`	пустая строка (`""`)
`DateTime`	`t"1970-01-01T00:00:00.000Z"`
`Enum`	`-1`
`IPv6`	`::`
`UInt`	`0`
`Array`	`[]`
`MAC`	пустая строка (`""`)

Пример 1224. Примеры преобразования значений к нужным типам

```
to_string(.dproc) ?? ""              # Преобразование к строке LCString
to_int(dvcpid) ?? 0                  # Преобразование к UInt
ip_to_ipv6(agent.ip) ?? "::"         # Преобразование к IPv6
```

Шаблонизация строк

Для формирования сообщений, генерируемых программой, рекомендуется использовать строковую шаблонизацию. Это позволяет улучшить читаемость кода и избежать ошибок, связанных с объединением строк вручную.

Вместо использования функции `join` для формирования строки, можно применить шаблонизированные строки, которые поддерживаются в VRL. В шаблонизированной строке переменные значения заключаются в фигурные скобки `{}`, и их значения автоматически подставляются в строку.

Пример 1225. Пример кода с использованием функции `join`:

```
...
on_correlate: !vrl |
  %state = .
  %state.msg = join(["На хосте", (to_string(.dvc) ?? "-"),
                     "пользователем", (to_string(.duser) ?? "-"),
                     "было зафиксировано изменение параметров критичной политики"],
                     separator: " ") ?? "-"
...
```

В данном примере функция `join` используется для объединения строк и переменных. Этот подход требует явного указания разделителей и может приводить к сложностям при добавлении или изменении текста.

Пример 1226. Пример кода с использованием шаблонизации:

```
...
on_correlate: !vrl |
  %state = .
  %state.msg = "На хосте {to_string(.dvc) ?? '-'} пользователем {to_string(.duser) ??
'-'} было зафиксировано изменение параметров критичной политики"
...
```

Использование шаблонизации позволяет подставлять значения переменных прямо в строку, что упрощает структуру кода и снижает вероятность ошибок.

Контроль дублирования кода

Дублирование фрагментов кода в правилах может затруднить их поддержку и привести к ошибкам при изменении кода. Избегайте повторения фрагментов или объединяйте их.

Пример 1227. Код с дублированием

```
...
on_correlate: !vrl |
  . |= compact({
    "name": %state.name,
    "dhost": %state.dhost,
    "dntdom": %state.dntdom,  # Дублирование
```

```
    "dvc": %state.dvc,
    "dntdom": %state.dntdom,  # Дублирование
    ...
  })
...
```

Пример 1228. Оптимизированный код

```
...
on_correlate: !vrl |
  . |= compact({
    "name": %state.name,
    "dhost": %state.dhost,
    "dntdom": %state.dntdom,
    "dvc": %state.dvc,
    ...
  })
...
```

Оформление кода

В этом разделе описаны требования к наименованию переменных, отступам и форматированию, которые должны быть соблюдены для улучшения читаемости кода и уменьшения вероятности ошибок в процессе разработки.

Названия переменных

Названия переменных должны быть логичными и отражать их содержание.

Пример 1229. Пример плохого наименования

```
...
on_event: !vrl |
  x = .source_ip
...
```

Пример 1230. Оптимизированный код

```
...
on_event: !vrl |
  source_ip_address = .source_ip
...
```

Отступы и форматирование

Отступы и форматирование должны быть аккуратными и последовательными, что упрощает чтение и понимание кода.

Пример 1231. Пример плохого форматирования

```
if contains(.msg, "error"){state.msg = "Ошибка обнаружена"}
```

Пример 1232. Оптимизированный код

```
if contains(.msg, "error") {
  state.msg = "Ошибка обнаружена"
}
```

GraphQL API

Об API

API системы, основанный на технологии GraphQL, представляет собой комплексный интерфейс для выполнения запросов и манипуляций с данными в рамках системы.

Об API-токенах

Взаимодействие с API осуществляется через HTTP-запросы методами POST для отправки GraphQL-запросов и мутаций. Каждый запрос требует наличия авторизационного токена в заголовке запроса. API-токен выпускается системой посредством функционала ботов. Бот — учетная запись, предназначенная для автоматизированного взаимодействия с системой через API-запросы. При создании бота, ему автоматически присваивается API-токен.

О GraphQL

GraphQL представляет собой язык запросов для API, обладающий следующими характеристиками:

- **Спецификация**: определяет правила валидации схемы на стороне сервера API, а также валидность вызовов клиента на основе этой схемы.
- **Строгая типизация**: схема GraphQL строго типизирована, описывая типы данных и отношения между объектами, что обеспечивает предсказуемость и точность ответов API.
- **Иерархичность**: структура запросов и ответов GraphQL отражает иерархическую структуру данных, позволяя клиентам получать именно те данные, которые необходимы, в одном запросе.
- **Уровень приложения**: GraphQL функционирует на уровне приложения, не завися от конкретных технологий хранения данных. Графическая структура запросов и ответов описывается в схеме API, определяющей объекты и их связи.

Начало работы

Для интеграции с API системы рекомендуется выполнение ряда действий, таких как подготовка окружения, получение токена и авторизация.

Подготовка и настройка окружения

Установите и настройте необходимые инструменты, которые поддерживают формирование и отправку GraphQL-запросов. Например, такие, как Postman, Insomnia или GraphQL IDE.

Также система включает GraphQL-песочницу — интерфейс, в котором можно просмотреть схему доступных GraphQL-запросов и выполнить их.

GraphQL-песочница по умолчанию недоступна. Чтобы открыть доступ к ней, обратитесь к разделу Открытие доступа к GraphQL-песочнице.

После открытия доступа GraphQL-песочница будет доступна из браузера по

следующему адресу:

```
<system_url>/api/graphql
```

Здесь:

- `<system_url>` — URL, по которому осуществляется доступ к веб-интерфейсу системы.

Аутентификация

Доступ к функциональности API системы требует аутентификации с использованием токена авторизации. Токен — это уникальный цифро-буквенный код, представленный в формате JWT (JSON Web Token), который включается в каждый запрос через HTTP заголовок `Authorization`.

Токен можно получить, создав учетную запись с типом Бот в веб-интерфейса системы.

Для выполнения запросов к API, необходимо включить следующие HTTP заголовки:

- `Authorization`: заголовок, содержащий токен авторизации в формате JWT (JSON Web Token). Токен предоставляет информацию о пользователе и его правах доступа.
- `x-tenant-id`: заголовок, указывающий идентификатор тенанта.

Токен передается в заголовке `Authorization` HTTP-запроса при обращении к API.

`Bearer` не является частью самого токена и добавляется в заголовок запроса в качестве префикса перед самим токеном для указания схемы аутентификации.

Пример 1233. Необходимые HTTP заголовки для выполнения запросов:

```
{
  "Authorization": "Bearer <ваш_токен_доступа>",
  "x-tenant-id": "00000000-0000-0000-0000-000000000000"
}
```

Где `<ваш_токен_доступа>` — это актуальный токен доступа, полученный после аутентификации в системе.

Пример 1234. Пример HTTP заголовков:

```
{
  "Authorization": "Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...,
  "x-tenant-id": "99311ef1-32a1-4b2d-9d38-7613be5e...."
}
```

Выполнение первого запроса

Пример запроса на получение детальной информации об элементе экспертизы:

```
query {
  expertise(id: "uniqueExpertiseId") {
    id
    sourceId
    folderId
    type
    name
    description
    draft
    author
    status
    tags
    version
    severity
    createdAt
    updatedAt
    __typename
  }
}
```

В этом запросе используется параметр:

- `id`: уникальный идентификатор элемента экспертизы, информацию о котором необходимо получить.

▼ Возвращаемый тип `Expertise`

Параметр	Описание
`id`	Уникальный идентификатор элемента экспертизы.
`sourceId`	Идентификатор исходного объекта, если применимо.
`folderId`	Идентификатор каталога, в который включен элемент экспертизы.
`type`	Тип элемента экспертизы.
`name`	Название элемента экспертизы.
`description`	Описание элемента экспертизы.
`draft`	Указывает, является ли объект черновиком.
`author`	Автор элемента экспертизы. Указывается автоматически на основе данных пользователя, выполнившего мутацию.
`status`	Статус элемента экспертизы.
`tags`	Массив тегов, ассоциированных с объектом.
`version`	Версия элемента экспертизы.
`severity`	Уровень серьезности элемента экспертизы.

Параметр	Описание
`createdAt`	Дата и время создания объекта.
`updatedAt`	Дата и время последнего обновления объекта.
`__typename`	Название типа объекта в системе GraphQL.

Выполнение запросов

Для выполнения запросов GraphQL, учитывая, что операции GraphQL состоят из многострочного JSON, рекомендуется использовать GraphQL Explorer. В качестве альтернативы можно использовать curl или любую другую HTTP-совместимую библиотеку. Все запросы GraphQL, будь то запросы (queries), мутации (mutations) или подписки (subscriptions) передаются в теле POST-запроса в формате JSON.

Пример использования curl для выполнения GraphQL запроса:

```
curl -H "Authorization: Bearer YOUR_ACCESS_TOKEN" -H "x-tenant-id: YOUR_TENANT_ID" -X
POST -d " \
 { \
   \"query\": \"query { yourQueryHere }\" \
 } \
" https://your-rvision-evo-domain/api/graphql
```

В этом примере:

- `YOUR_ACCESS_TOKEN` должен быть заменен на ваш фактический токен доступа к системе.
- `YOUR_TENANT_ID` должен быть заменен на идентификатор вашего тенанта в системе.
- `yourQueryHere` должен быть заменен на ваш конкретный запрос GraphQL, например, запрос на получение списка событий или информации о специфическом объекте.
- URL `https://your-rvision-evo-domain/api/graphql` представляет собой конечную точку для доступа к GraphQL API вашей системы и должен быть заменен на актуальный URL вашего API.

Важно корректно экранировать специальные символы в теле JSON-запроса для обеспечения его правильной обработки сервером. Внешние двойные кавычки и экранированные внутренние двойные кавычки помогают избежать ошибок при разборе запроса.

При отсутствии необходимых разрешений у токена для доступа к запрашиваемому ресурсу, API вернет ошибку с указанием, какие разрешения необходимы для выполнения запроса.

Открытие доступа к GraphQL-песочнице

Чтобы открыть доступ к GraphQL-песочнице, выполните следующие действия:

1. Подключитесь к терминалу управляющего узла кластера Kubernetes, в котором установлена система.

2. Откройте карту конфигурации `evo.siem-bff-env`, выполнив следующую команду:

   ```
   kubectl -n <namespace> edit configmap evo.siem-bff-env
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.
3. Установите для переменной `SIEM_BFF_GRAPHQL_ENABLE_PLAYGROUND` значение `"true"`.
4. Сохраните изменения в карте конфигурации.
5. Перезапустите сервис `bff`, выполнив следующую команду:

   ```
   kubectl -n <namespace> rollout restart deployment/evo.siem.bff
   ```

   Здесь:

   - `<namespace>` — имя пространства имен, в котором установлен кластер.

После открытия доступа GraphQL-песочница будет доступна из браузера по следующему адресу:

```
<system_url>/api/graphql
```

Здесь:

- `<system_url>` — URL, по которому осуществляется доступ к веб-интерфейсу системы.

API-токены

Данный раздел описывает процесс работы с API-токенами в системе.

Об API-токенах

API системы обеспечивает программный доступ к функционалу системы через протокол HTTP. Доступные операции API включают выполнение запросов и мутаций с использованием GraphQL, что позволяет управлять данными и настройками системы.

Взаимодействие с API осуществляется через HTTP-запросы методами POST для отправки GraphQL-запросов и мутаций. Каждый запрос требует наличия авторизационного токена в заголовке запроса. API-токен выпускается системой посредством функционала ботов. Бот — учетная запись, предназначенная для автоматизированного взаимодействия с системой через API-запросы. При создании бота, ему автоматически присваивается API-токен.

Работа с API-токенами

Доступные операции над API-токенами:

- Создание API-токена

- Копирование API-токена
- Перевыпуск API-токена
- Удаление API-токена

Создание API-токена

Чтобы выпустить API-токен, необходимо создать учетную запись типа **Бот**. При создании бота, ему автоматически присваивается API-токен, который можно использовать для создания запросов к системе.

Чтобы создать бота:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Нажмите на кнопку **Добавить** (+) и выберите из выпадающего списка тип **Бот**. Система отобразит окно добавления бота.
3. Заполните следующие поля:
   - **Логин** — задайте уникальный логин для учетной записи пользователя в системе.
   - **Группа пользователей** (опционально) — выберите из выпадающего списка группы, в которые должен входить пользователь.
   - **Описание** (опционально) — добавьте краткое описание учетной записи пользователя.
4. Назначьте роли пользователю, чтобы определить перечень его разрешений в системе.

   Чтобы добавить роль:

   a. Нажмите на кнопку **Добавить роль** (+). Появится строка настройки роли.

   b. Выберите из выпадающего списка роль, которую необходимо назначить пользователю.

   c. При необходимости выберите из выпадающего списка тенанты, в рамках которых должна действовать роль.

   > Если тенанты не выбраны, роль действует во всех тенантах.

   Чтобы удалить добавленную роль, нажмите на кнопку в строке настройки роли.

   > Если активирован переключатель **Суперадминистратор**, учетной записи предоставляются все доступные в системе разрешения.

5. Нажмите на кнопку **Добавить**. Система создаст бота и отобразит соответствующее уведомление. Новый бот появится в списке раздела.

Копирование API-токена

Чтобы скопировать API-токен:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит список доступных пользователей.

2. Нажмите на строку нужного бота в списке. Система отобразит в правой части экрана карточку этого пользователя с подробной информацией о нем.
3. Нажмите на кнопку **Скопировать токен** в нижней части карточки. Система скопирует выбранный токен в буфер обмена и отобразит соответствующее уведомление.

Перевыпуск API-токена

Чтобы сбросить период хранения токена, необходимо перевыпустить токен для бота, к которому относится нужный токен. Предыдущий токен будет удален и станет недоступным для использования. Срок жизни токена будет отсчитываться от даты перевыпуска API-токена.

Чтобы обновить API-токен:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит список доступных пользователей.
2. Нажмите на строку нужного бота в списке. Система отобразит в правой части экрана карточку этого бота с подробной информацией о нем.
3. Нажмите на кнопку **Перевыпустить токен** в нижней части карточки. Система выпустит новый API-токен и отобразит соответствующее уведомление. Предыдущий токен бота станет недействительным.

Удаление API-токена

Чтобы удалить API-токен, необходимо удалить связанного с ним бота:

1. Перейдите в раздел **Настройки** → **Управление доступом** → **Пользователи**. Система отобразит сведения о доступных пользователях.
2. Наведите курсор на строку нужного бота в списке. Нажмите на кнопку **Удалить** (), которая отображается в правой части строки. Отобразится окно подтверждения удаления бота.
3. Нажмите на кнопку **Удалить**. Система удалит бота и отобразит соответствующее уведомление. Удаленный бот будет исключен из списка пользователей. API-токен, связанный с ботом, станет недействительным.

Работа с GraphQL

В данном разделе представлено описание функционала выполнения запросов, мутаций и подписок для взаимодействия с данными системы через *GraphQL API*.

API позволяет выполнять следующие операции:

- Выгрузка событий: методы для извлечения данных о событиях, поддерживающие фильтрацию, сортировку и пагинацию.
- Управление элементами экспертизы: создание, обновление, публикация и удаление элементов экспертизы.
- Просмотр и управление каталогами: запросы для получения списка каталогов и управления ими.
- Фильтры и сортировка: применение фильтров для уточнения запросов и настройка порядка сортировки результатов.

- Пагинация: методы для организации данных в пагинированном виде.

Выполнение запросов

Для выполнения запросов GraphQL, учитывая, что операции GraphQL состоят из многострочного JSON, рекомендуется использовать GraphQL Explorer. В качестве альтернативы можно использовать curl или любую другую HTTP-совместимую библиотеку. Все запросы GraphQL, будь то запросы (queries), мутации (mutations) или подписки (subscriptions) передаются в теле POST-запроса в формате JSON.

Пример использования curl для выполнения GraphQL запроса:

```
curl -H "Authorization: Bearer YOUR_ACCESS_TOKEN" -H "x-tenant-id: YOUR_TENANT_ID" -X
POST -d " \
 { \
   \"query\": \"query { yourQueryHere }\" \
 } \
" https://your-rvision-evo-domain/api/graphql
```

В этом примере:

- `YOUR_ACCESS_TOKEN` должен быть заменен на ваш фактический токен доступа к системе.
- `YOUR_TENANT_ID` должен быть заменен на идентификатор вашего тенанта в системе.
- `yourQueryHere` должен быть заменен на ваш конкретный запрос GraphQL, например, запрос на получение списка событий или информации о специфическом объекте.
- URL `https://your-rvision-evo-domain/api/graphql` представляет собой конечную точку для доступа к GraphQL API вашей системы и должен быть заменен на актуальный URL вашего API.

Важно корректно экранировать специальные символы в теле JSON-запроса для обеспечения его правильной обработки сервером. Внешние двойные кавычки и экранированные внутренние двойные кавычки помогают избежать ошибок при разборе запроса.

При отсутствии необходимых разрешений у токена для доступа к запрашиваемому ресурсу, API вернет ошибку с указанием, какие разрешения необходимы для выполнения запроса.

Выгрузка событий

Позволяет осуществлять выгрузку событий через подписку `exportSearch`, поддерживая фильтрацию, сортировку и пагинацию.

Пример 1235. Структура подписки `exportSearch`

```
subscription {
  exportSearch(
    eventStorageId: String!
    filters: [SearchFilter!]
    limit: Int
```

```
    rql: String!
    timePeriod: TimePeriodInput!
  ) {
    __typename
    searchId
    exportResult {
      ... on ExportSearchLimitWarning {
        limitWarning
        __typename
      }
      ... on ExportSearchResult {
        result {
          ... on SearchResultHeaders {
            names
            types
            __typename
          }
          ... on SearchResultRows {
            rows
            __typename
          }
          __typename
        }
        __typename
      }
      __typename
    }
  }
}
```

Таблица 42. Параметры подписки `exportSearch`

Параметр	Тип данных	Обязателен	Описание
`eventStorageId`	Строка	Да	Идентификатор хранилища событий, указывает источник экспортируемых данных.
`filters`	Массив `[SearchFilter!]`	Да	Список фильтров, применяемых к выгрузке.
`limit`	Целое число	Нет	Ограничение на количество возвращаемых записей.
`rql`	Строка	Да	RQL-запрос для фильтрации данных.
`timePeriod`	Объект `TimePeriodInput!`	Да	Временной интервал для выгрузки данных.

Рассмотрим структуру подписки:

- `subscription`: указывает на тип операции подписки в GraphQL, позволяющее получать данные в реальном времени.
- `exportSearch`: имя подписки, используемой для выгрузки данных событий.
- `eventStorageId`: обязательный параметр, принимающий идентификатор хранилища событий, из которого будут экспортированы данные.
- `filters`: массив фильтров, применяемых к выгрузке. Каждый фильтр задается объектом `SearchFilter`, который включает поля `inverted`, `name`, `operation`, и `value`.
- `limit`: ограничение на количество возвращаемых записей. Необязательный параметр.
- `rql`: строка RQL-запроса, используемая для дополнительной фильтрации выгружаемых данных.
- `timePeriod`: временной интервал, за который происходит выгрузка данных. Принимает объект `TimePeriodInput` с полями `from` и `to`, указывающими начало и конец интервала соответственно.

В ответ на подписку возвращаются следующие данные:

- `searchId`: идентификатор созданного поиска, по которому происходит выгрузка данных.
- `exportResult`: объект, содержащий результат выгрузки. Может представлять предупреждение о превышении лимита `ExportSearchLimitWarning` или непосредственно результаты поиска `ExportSearchResult`.
- `ExportSearchLimitWarning` содержит поле `limitWarning`, информирующее о превышении установленного лимита записей.
- `ExportSearchResult` включает в себя `result`, который может быть типом `SearchResultHeaders` с информацией о заголовках и типах данных или `SearchResultRows` с собственно данными строк.

Пример 1236. Пример подписки `exportSearch`

```
subscription {
  exportSearch(
    eventStorageId: "34b1b60a-9cce-4a64-b69a-5837a433b730",
    filters: [
      {
        inverted: false,
        name: "collectorId",
        operation: EQ,
        value: "'a7e7a9c3-c69e-460e-967a-1e437601fd71'"
      }
    ],
    limit: 500,
    rql: "",
    timePeriod: {
      from: "2024-02-14T07:19:33.806Z",
      to: "2024-02-14T07:20:33.806Z"
    }
  ) {
    __typename
    searchId
```

```
    exportResult {
      ... on ExportSearchLimitWarning {
        limitWarning
        __typename
      }
      ... on ExportSearchResult {
        result {
          ... on SearchResultHeaders {
            names
            types
            __typename
          }
          ... on SearchResultRows {
            rows
            __typename
          }
          __typename
        }
        __typename
      }
      __typename
    }
  }
}
```

Как работает данная подписка:

- `subscription`: определяет начало подписки, которая будет активно слушать изменения и отправлять данные клиенту, как только они станут доступны.
- `exportSearch`: имя подписки, которое инициирует процесс выгрузки данных событий из системы.
- `eventStorageId`: идентификатор хранилища событий, который указывает, откуда будут экспортироваться данные. В данном случае "34b1b60a-9cce-4a64-b69a-5837a433b730".
- `filters`: список фильтров, применяемых к выгрузке. В примере используется один фильтр для поля "collectorId" с операцией сравнения EQ (равно) и значением "'a7e7a9c3-c69e-460e-967a-1e437601fd71'". Фильтр не инвертирован (`inverted: false`), что означает прямое сравнение.
- `limit`: ограничивает количество возвращаемых записей до 500.
- `rql`: параметр, представляющий RQL-запрос для дополнительной фильтрации данных. В данном случае он пуст, что означает отсутствие дополнительных условий фильтрации.
- `timePeriod`: задает временной интервал для выгрузки данных, начиная с "2024-02-14T07:19:33.806Z" и заканчивая "2024-02-14T07:20:33.806Z".
- `searchId`: возвращаемый идентификатор нового поиска, который может использоваться для отслеживания процесса выгрузки.
- `exportResult`: объект, содержащий результат выгрузки. Используется фрагмент `ExportSearchResult`, который может представлять либо предупреждение о превышении лимита (`ExportSearchLimitWarning`), либо непосредственно результаты поиска (`ExportSearchResult`).

Параметры `ExportSearchResult` и `SearchResult` определяют структуру возвращаемых данных:

- `ExportSearchLimitWarning` содержит `limitWarning`, информирующее о превышении заданного лимита выгрузки.
- `ExportSearchResult` включает `result`, который далее разделяется на `SearchResultHeaders` с информацией о заголовках и типах данных и `SearchResultRows` с данными строк.

`__typename` используется для определения конкретного типа возвращаемого объекта в ответе, что позволяет клиенту корректно обрабатывать различные типы результатов выгрузки.

Сохранение черновика элемента экспертизы

Мутация `saveExpertiseDraft` позволяет сохранить черновик элемента экспертизы в системе.

Пример 1237. Структура мутации `saveExpertiseDraft`

```
mutation {
  saveExpertiseDraft(input: SaveExpertiseDraftInput!) {
    id
    sourceId
    folderId
    type
    name
    description
    draft
    author
    status
    tags
    version
    severity
    createdAt
    updatedAt
    __typename
  }
}
```

Таблица 43. Параметры мутации `saveExpertiseDraft`

Параметр	Тип данных	Обязателен	Описание
`folderId`	Строка	Да	Идентификатор каталога, в который добавляется черновик.
`id`	Строка	Нет	Идентификатор существующего элемента экспертизы для обновления.
`sourceCode`	Строка	Да	Исходный код элемента экспертизы.

Параметр	Тип данных	Обязателен	Описание
type	ExpertiseType	Да	Тип элемента экспертизы, определяющий его категорию или классификацию.

▼ *Возвращаемый тип* Expertise

Параметр	Описание
id	Уникальный идентификатор элемента экспертизы, генерируемый системой.
sourceId	Идентификатор источника данных для элемента экспертизы.
folderId	Идентификатор каталога, к которому принадлежит объект.
type	Тип элемента экспертизы.
name	Название элемента экспертизы.
description	Описание элемента экспертизы.
draft	Статус черновика объекта.
author	Автор элемента экспертизы.
status	Текущий статус элемента экспертизы.
tags	Массив тегов, ассоциированных с объектом.
version	Версия элемента экспертизы.
severity	Уровень серьезности элемента экспертизы.
createdAt	Дата и время создания объекта.
updatedAt	Дата и время последнего обновления объекта.
__typename	Тип объекта в системе GraphQL.

Пример 1238. Пример мутации saveExpertiseDraft

```
mutation {
  saveExpertiseDraft(input: {
    folderId: "1de2e868-768c-4707-bb5e-7de4413f833f",
    id: "01c3b9ef-b1fa-4ff9-a465-c89d43cb537b",
    sourceCode: "Вставьте исходный код элемента экспертизы",
    type: "CORRELATION_RULE"
  }) {
    id
    sourceId
    folderId
    type
    name
    description
    draft
    author
```

```
    status
    tags
    version
    severity
    createdAt
    updatedAt
    __typename
  }
}
```

Как работает данная мутация:

- `mutation`: указывает на начало операции мутации.
- `saveExpertiseDraft`: имя мутации, используемой для сохранения черновика элемента экспертизы.
- `input`: параметр, который принимает данные для сохранения черновика. Эти данные упакованы в объект `SaveExpertiseDraftInput`.
- `SaveExpertiseDraftInput`: входной объект мутации, содержащий необходимые данные для создания или обновления черновика элемента экспертизы. Включает в себя поля, такие как `folderId`, `id`, `sourceCode`, и `type`.
- `folderId`: идентификатор каталога, в который добавляется черновик. Этот параметр обязателен.
- `id`: необязательный параметр, указывающий идентификатор существующего элемента экспертизы, если требуется обновление черновика.
- `sourceCode`: текстовое содержание кода элемента экспертизы. Обязательный параметр.
- `type`: тип элемента экспертизы, определяющий его функциональную принадлежность или категорию. Этот параметр обязателен.

Возвращаемые поля мутации `saveExpertiseDraft`:

- `id`: уникальный идентификатор созданного или обновленного элемента экспертизы.
- `sourceId`, `folderId`, `type`, `name`, `description`: основные атрибуты элемента экспертизы, включая его исходный идентификатор, принадлежность к каталогу, тип, название, и описание.
- `draft`: статус черновика объекта, указывающий, находится ли объект в разработке.
- `author`: автор элемента экспертизы.
- `status`, `tags`, `version`, `severity`: дополнительные атрибуты объекта, такие как его статус, теги, версия и уровень серьезности.
- `createdAt`, `updatedAt`: даты создания и последнего обновления элемента экспертизы.
- `__typename`: служебное поле GraphQL, указывающее тип возвращаемого объекта.

Публикация версии элемента экспертизы

Мутация `publishExpertise` позволяет публиковать версии элементов экспертизы, переводя их из статуса "Черновик" в статус "Опубликовано". Опубликованные объекты становятся доступными для просмотра и использования в рамках системы.

Пример 1239. Структура мутации `publishExpertise`

```
mutation {
  publishExpertise(input: PublishExpertiseInput!) {
    id
    sourceId
    folderId
    type
    name
    description
    draft
    author
    status
    tags
    version
    severity
    createdAt
    updatedAt
    __typename
  }
}
```

Таблица 44. Параметры мутации `publishExpertise`

Параметр	Тип данных	Обязателен	Описание
`folderId`	Строка	Да	Идентификатор каталога, к которому будет принадлежать опубликованный объект.
`id`	Строка	Нет	Идентификатор существующего черновика элемента экспертизы.
`sourceCode`	Строка	Да	Исходный код элемента экспертизы для публикации.
`type`	`ExpertiseType`	Да	Тип элемента экспертизы, указывающий на его функциональную категорию.

▼ *Возвращаемый тип `Expertise`*

Параметр	Описание
`id`	Уникальный идентификатор опубликованного элемента экспертизы.
`sourceId`	Идентификатор источника данных для элемента экспертизы.

Параметр	Описание
`folderId`	Идентификатор каталога, к которому принадлежит опубликованный объект.
`type`	Тип опубликованного элемента экспертизы.
`name`	Название опубликованного элемента экспертизы.
`description`	Описание опубликованного элемента экспертизы.
`draft`	Статус черновика, указывающий на стадию разработки объекта.
`author`	Автор элемента экспертизы.
`status`	Текущий статус опубликованного элемента экспертизы.
`tags`	Массив тегов, ассоциированных с элементом экспертизы.
`version`	Версия элемента экспертизы.
`severity`	Уровень серьезности элемента экспертизы.
`createdAt`	Дата и время создания элемента экспертизы.
`updatedAt`	Дата и время последнего обновления элемента экспертизы.
`__typename`	Тип объекта в системе GraphQL.

Пример 1240. Пример мутации `publishExpertise`

```
mutation {
  publishExpertise(input: {
    folderId: "1de2e868-768c-4707-bb5e-7de4413f833f",
    id: "01c3b9ef-b1fa-4ff9-a465-c89d43cb537b",
    sourceCode: "Вставьте исходный код элемента экспертизы",
    type: CORRELATION_RULE
  }) {
    id
    sourceId
    folderId
    type
    name
    description
    draft
    author
    status
    tags
    version
    severity
    createdAt
    updatedAt
    __typename
  }
}
```

Как работает данная мутация:

- `mutation`: указывает на начало операции мутации.
- `publishExpertise`: имя мутации для публикации элемента экспертизы.
- `input`: параметр, принимающий данные для публикации черновика. Включает в себя `folderId`, `id` (если это обновление существующего объекта), `sourceCode`, и `type`.
- `folderId`: идентификатор каталога для нового элемента экспертизы.
- `id`: необязательный параметр, указывающий на существующий элемент экспертизы.
- `sourceCode`: исходный код элемента экспертизы.
- `type`: тип элемента экспертизы, определяющий его функциональную категорию.

Возвращаемые поля мутации `publishExpertise`:

- `id`: уникальный идентификатор опубликованного элемента экспертизы.
- `sourceId`, `folderId`, `type`, `name`, `description`: основные атрибуты элемента экспертизы, включая его исходный идентификатор, принадлежность к каталогу, тип, название и описание.
- `draft`: статус объекта.
- `author`: автор элемента экспертизы.
- `status`, `tags`, `version`, `severity`: дополнительные атрибуты объекта, такие как его статус, теги, версия и уровень серьезности.
- `createdAt`, `updatedAt`: даты создания и последнего обновления элемента экспертизы.
- `__typename`: служебное поле GraphQL, указывающее тип возвращаемого объекта.

Экспорт элементов экспертизы

Запрос `exportExpertise` предоставляет возможность экспортировать элементы экспертизы, определяемые их уникальными идентификаторами (UUID элемента экспертизы), позволяя пользователям системы сохранить эти данные вне системы.

Пример 1241. Структура запроса `exportExpertise`

```
exportExpertise(ids: [String!]!) {
    content
    contentType
  }
}
```

Таблица 45. Параметры запроса `exportExpertise`

Параметр	Тип данных	Обязателен	Описание
`ids`	Массив `[ID!]`	Да	Массив значений UUID элементов экспертизы для экспорта.

▼ *Возвращаемый тип `ExportExpertiseResponse`*

Параметр	Описание
`content`	Содержимое экспортированных элементов экспертизы в формате base64. Клиент должен декодировать это содержимое и сохранить как файл в формате ZIP.
`contentType`	Тип содержимого, обычно указывается как `application/zip`, обозначает формат возвращаемого файла экспорта.

Пример 1242. Пример запроса `exportExpertise`

```
query {
  exportExpertise(ids: ["uuid1", "uuid2"]) {
    content
    contentType
  }
}
```

Этот запрос инициирует экспорт элементов экспертизы с указанными UUID. В ответе предоставляется закодированное содержимое экспортированных объектов в формате base64 и тип содержимого, что позволяет клиенту обработать и сохранить данные в виде ZIP-файла.

Как работает данный запрос:

- `query`: указывает на начало операции запроса.
- `exportExpertise`: имя запроса, используемого для экспорта элементов экспертизы.
- `ids`: параметр запроса, представляющий собой массив UUID элементов экспертизы, которые будут экспортированы.

Возвращаемые поля мутации `exportExpertise`:

- `content`: содержимое экспортированных объектов в формате base64, позволяет пользователям декодировать и сохранить информацию в виде файла.
- `contentType`: тип содержимого экспорта, указывающий на формат файла (например, `application/zip` для ZIP-архивов).

Просмотр каталогов

Запрос `resourceFolders` предлагает обзор всех каталогов, содержащих элементы экспертизы в системе.

Пример 1243. Структура запроса `resourceFolders`

```
query {
  resourceFolders {
    id
    name
    parentFolderId
    createdAt
    updatedAt
```

```
    creator {
      id
      name
    }
    updater {
      id
      name
    }
  }
}
```

▼ *Возвращаемый тип* `ResourceFolder`

Параметр	Описание
`id`	Уникальный идентификатор каталога.
`name`	Название каталога.
`parentFolderId`	Идентификатор родительского каталога.
`createdAt`	Дата и время создания каталога.
`updatedAt`	Дата и время последнего обновления каталога.
`creator`	Информация о пользователе, создавшем каталог.
`updater`	Информация о последнем пользователе, обновившем каталог.

Пример 1244. Пример запроса на получение списка каталогов

```
query {
  resourceFolders {
    id
    name
    parentFolderId
    createdAt
    updatedAt
    creator {
      name
    }
    updater {
      name
    }
  }
}
```

Данный запрос возвращает иерархический список всех каталогов, доступных в системе, включая информацию об их создании и последних изменениях.

Просмотр элемента экспертизы

Запрос `expertise` предоставляет детальную информацию о конкретном элементе экспертизы, идентифицируемом по его уникальному идентификатору в системе.

Пример 1245. Структура запроса `expertise`

```
query {
  expertise(id: ID!) {
    id
    sourceId
    folderId
    type
    name
    description
    draft
    author
    status
    tags
    version
    severity
    createdAt
    updatedAt
    __typename
  }
}
```

Таблица 46. Параметры запроса `expertise`

Параметр	Тип данных	Обязателен	Описание
`id`	ID!	Да	Уникальный идентификатор элемента экспертизы для получения информации.

▼ *Возвращаемый тип `Expertise`*

Параметр	Описание
`id`	Уникальный идентификатор элемента экспертизы.
`sourceId`	Идентификатор исходного объекта, если применимо.
`folderId`	Идентификатор каталога, содержащего элемент экспертизы.
`type`	Тип элемента экспертизы, указывающий на его функциональную категорию.
`name`	Название элемента экспертизы.
`description`	Описание объекта, предоставляющее детальную информацию о его содержании.
`draft`	Статус черновика, указывающий на текущее состояние объекта.
`author`	Автор элемента экспертизы.
`status`	Текущий статус объекта в системе.
`tags`	Массив тегов, ассоциированных с объектом.

Параметр	Описание
`version`	Версия элемента экспертизы, отражающая его историю изменений.
`severity`	Уровень серьезности объекта, определяющий его важность.
`createdAt`	Дата и время создания объекта.
`updatedAt`	Дата и время последнего обновления объекта.
`__typename`	Служебное поле GraphQL, указывающее тип возвращаемого объекта.

Просмотр списка элементов экспертизы

Запрос `expertises` обеспечивает возможность получения списка элементов экспертизы в системе, поддерживая фильтрацию, сортировку и пагинацию для удобства навигации и анализа.

Пример 1246. Структура запроса `expertises`

```
query {
  expertises(
    filter: ExpertisesFilterInput,
    pagination: PaginationInput,
    sorting: SortingInput
  ) {
    expertises {
      id
      sourceId
      folderId
      type
      name
      description
      draft
      author
      status
      tags
      version
      severity
      createdAt
      updatedAt
      __typename
    }
    pageInfo {
      pagination {
        limit
        offset
        __typename
      }
      totalCount
      __typename
    }
    __typename
  }
```

```
}
```

Таблица 47. Параметры запроса `expertises`

Параметр	Тип данных	Обязателен	Описание
`filter`	`ExpertisesFilterInput`	Нет	Параметры фильтрации для настройки списка элементов экспертизы.
`pagination`	`PaginationInput`	Нет	Параметры пагинации для контроля количества отображаемых результатов.
`sorting`	`SortingInput`	Нет	Параметры сортировки для упорядочивания результатов запроса.

▼ *Возвращаемый тип* `Expertises`

Параметр	Описание
`expertises`	Список элементов экспертизы, соответствующих заданным критериям фильтрации, сортировки и пагинации.
`pageInfo`	Метаинформация о странице, включая общее количество доступных объектов и параметры текущей пагинации.
`totalCount`	Общее количество элементов экспертизы, соответствующих критериям фильтрации.
`__typename`	Служебное поле GraphQL, указывающее тип возвращаемого объекта.

Пример 1247. Пример запроса на получение списка элементов экспертизы

```
query {
  expertises(
    filter: {
      folder: {
        folderId: "1de2e868-768c-4707-bb5e-7de4413f833f",
        withChildFolders: true
      },
      query: "",
      status: ENABLED,
      type: CORRELATION_RULE
    },
    pagination: {
      limit: 100,
      offset: 0
    },
    sorting: {
      order: DESC,
      orderBy: CREATED_AT
```

```
    }
  ) {
    expertises {
      id
      sourceId
      folderId
      type
      name
      description
      draft
      author
      status
      tags
      version
      severity
      createdAt
      updatedAt
      __typename
    }
    pageInfo {
      pagination {
        limit
        offset
        __typename
      }
      totalCount
      __typename
    }
    __typename
  }
}
```

Этот запрос инициирует вывод списка элементов экспертизы, применяя заданные критерии фильтрации, пагинации и сортировки.

Как работает данный запрос:

- `query`: указывает на начало операции запроса.
- `expertises`: имя запроса, используемое для получения списка элементов экспертизы.
- `filter`: параметр, принимающий критерии фильтрации для применения к запросу. Включает:
  - `folder`: объект с `folderId`, указывающий на конкретный каталог, и `withChildFolders`, указывающий, следует ли включать объекты из дочерних каталогов.
  - `query`: строка для поиска по элементам экспертизы, в данном примере пустая, что означает отсутствие фильтрации по текстовому запросу.
  - `status`: фильтр по статусу объектов, в данном случае `ENABLED`, что указывает на выборку активных объектов.
  - `type`: фильтр по типу объекта, в данном случае `CORRELATION_RULE`, ограничивающий выборку объектами определенного типа.

- `pagination`: объект с параметрами пагинации, `limit` устанавливает максимальное количество объектов в ответе, `offset` — смещение от начала списка.
- `sorting`: параметры сортировки результатов запроса, в данном случае сортировка по дате создания (`CREATED_AT`) в обратном порядке (`DESC`).

Возвращаемые данные запроса `expertises`:

- `expertises`: массив элементов экспертизы, каждый из которых содержит:
  - `id`, `sourceId`, `folderId`, `type`, `name`, `description`, `draft`, `author`, `status`, `tags`, `version`, `severity`, `createdAt`, `updatedAt`: основные атрибуты элемента экспертизы.
  - `__typename`: служебное поле GraphQL, указывающее тип возвращаемого объекта.
- `pageInfo`: объект с информацией о пагинации и общим количеством объектов в ответе, включает:
  - `pagination`: внутренний объект с текущими параметрами пагинации (`limit`, `offset`).
  - `totalCount`: общее количество доступных элементов экспертизы с учетом примененных фильтров.
  - `__typename`: служебное поле для объекта пагинации и общего количества.

Удаление элемента экспертизы

Мутация `deleteExpertise` позволяет удалить элемент экспертизы из системы.

```
deleteExpertise(input: DeleteExpertiseInput!): DeleteExpertiseResponse!
```

Параметр	Тип данных	Обязателен	Описание
`input`	`DeleteExpertiseInput`	Да	Входные данные для удаления, включая `id` удаляемого элемента экспертизы.

Таблица 48. Параметры мутации `DeleteExpertiseInput`

Параметр	Тип данных	Обязателен	Описание
`id`	Строка	Да	Идентификатор элемента экспертизы, который требуется удалить.

Таблица 49. Возвращаемый тип `DeleteExpertiseResponse`

Параметр	Тип данных	Обязателен	Описание
`id`	Строка	Да	Идентификатор удаленного элемента экспертизы, подтверждающий успешное выполнение операции.

Пример использования мутации для удаления экспертизы:

```
mutation {
  deleteExpertise(input: { id: "uniqueExpertiseId" }) {
    id
  }
}
```

Этот запрос инициирует удаление элемента экспертизы с указанным идентификатором из системы. В ответе возвращается идентификатор удаленного элемента экспертизы.

Фильтры

GraphQL API системы поддерживает использование фильтров для уточненного поиска и отбора данных, возвращаемых при выполнении запросов или подписок. Они могут быть применены к различным полям и поддерживают разнообразные операции сравнения.

Использование фильтров

Для применения фильтров в запросах или подписках, используется параметр `filters`. Каждый фильтр определяет `name` поля для фильтрации, `operation` (тип операции), `value` (значение для сравнения) и, опционально, `inverted` (инвертирование условия).

Пример 1248. Пример запроса с использованием фильтров

```
query {
  expertises(
    filter: {
      folder: {
        folderId: "1de2e868-768c-4707-bb5e-7de4413f833f",
        withChildFolders: true
      },
      query: "",
      status: ENABLED
      type: CORRELATION_RULE
    },
    pagination: {
      limit: 100,
      offset: 0
    },
    sorting: {
      order: DESC,
      orderBy: CREATED_AT
    }
    ) {
    expertises {
      id
      sourceId
      folderId
      type
```

```
      name
      description
      draft
      author
      status
      tags
      version
      severity
      createdAt
      updatedAt
      __typename
    }
    pageInfo {
      pagination {
        limit
        offset
        __typename
      }
      totalCount
      __typename
    }
    __typename
  }
}
```

Операции сравнения

В фильтрах используются следующие операции сравнения:

- `BETWEEN` — между двумя значениями;
- `EQ` — равно;
- `GE` — больше или равно;
- `GT` — больше;
- `IN` — входит в перечень;
- `LE` — меньше или равно;
- `LIKE` — соответствует шаблону;
- `LT` — меньше;
- `NE` — не равно.

Набор доступных операторов может варьироваться в зависимости от запроса.

Параметры фильтра

Таблица 50. Параметры фильтра

Параметр	Тип	Описание
`inverted`	Логический	Определяет, будет ли условие инвертировано.
`name`	Строка	Имя поля, по которому применяется фильтр.
`operation`	Перечисление	Операция сравнения, используемая в фильтре.
`value`	Строка	Значение для сравнения с полем.

Сортировка и пагинация

В GraphQL API системы предусмотрены механизмы сортировки и пагинации, позволяющие управлять порядком представления данных и их разбивкой на удобные для восприятия части. Эти механизмы особенно важны при работе с запросами, возвращающими большое количество записей, например, `expertises`.

Сортировка

Сортировка в запросах GraphQL API системы позволяет упорядочить результаты запроса по определенным критериям. В запросе `expertises` параметр `sorting` используется для указания направления сортировки (`order`) и поля, по которому следует сортировать (`orderBy`).

Пример ниже демонстрирует, как можно запросить список экспертиз, отсортированный по дате создания (`CREATED_AT`) в убывающем порядке (`DESC`), с применением фильтра и пагинации:

Пример 1249. Пример запроса с сортировкой

```
query {
  expertises(
    filter: {
      folder: {
        folderId: "1de2e868-768c-4707-bb5e-7de4413f833f",
        withChildFolders: true
      },
      query: "",
      status: ENABLED,
      type: CORRELATION_RULE
    },
    pagination: {
      limit: 100,
      offset: 0
    },
    sorting: {
      order: DESC,
      orderBy: CREATED_AT
    }
  ) {
    expertises {
```

```
      id
      sourceId
      folderId
      type
      name
      description
      draft
      author
      status
      tags
      version
      severity
      createdAt
      updatedAt
      __typename
    }
    pageInfo {
      pagination {
        limit
        offset
        __typename
      }
      totalCount
      __typename
    }
    __typename
  }
}
```

Таблица 51. Параметры сортировки

Параметр	Тип	Описание
`order`	Перечисление	Направление сортировки, может быть `ASC` (по возрастанию) или `DESC` (по убыванию).
`orderBy`	Строка	Поле, по которому будет проводиться сортировка.

Пагинация

Пагинация в GraphQL API системы предоставляет механизм для контроля объема данных, возвращаемых в результате запроса при работе с большими наборами данных, позволяя разбивать их на управляемые "страницы".

Пример запроса ниже иллюстрирует, как получить список элементов экспертизы с использованием пагинации для ограничения количества возвращаемых записей и управления точкой начала выборки данных:

Пример 1250. Пример запроса с пагинацией

```
query {
  expertises(
    filter: {
      folder: {
        folderId: "1de2e868-768c-4707-bb5e-7de4413f833f",
        withChildFolders: true
      },
      query: "",
      status: ENABLED,
      type: CORRELATION_RULE
    },
    pagination: {
      limit: 100,
      offset: 0
    },
    sorting: {
      order: DESC,
      orderBy: CREATED_AT
    }
    ) {
    expertises {
      id
      sourceId
      folderId
      type
      name
      description
      draft
      author
      status
      tags
      version
      severity
      createdAt
      updatedAt
      __typename
    }
    pageInfo {
      pagination {
        limit
        offset
        __typename
      }
      totalCount
      __typename
    }
    __typename
  }
```

```
}
```

Таблица 52. Параметры пагинации

Параметр	Тип	Описание
`limit`	Целое число	Максимальное количество записей, возвращаемых за один запрос.
`offset`	Целое число	Смещение от начала списка записей, с которого начинается возвращение данных.

Запросы API модуля управления уязвимостями

В данном разделе представлено описание функционала выполнения запросов и мутаций для взаимодействия с данными модуля управления уязвимостями через GraphQL API и примеры его использования.

В запросы к API необходимо включать HTTP-заголовок `x-tenant-id`, указывающий идентификатор тенанта.

Доступные операции:

- авторизация и получение токена.
- получение списка пространств.
- получение списка тенантов.
- получение списка сканеров.
- получение задачи сканирования, списка задач сканирования, истории запусков задачи сканирования.
- создание, редактирование, удаление, запуск, отмена запущенной задачи сканирования.
- приостановка и возобновление задачи сканироввния.
- получение профиля, списка профилей и списка стандартов.
- создание, редактирование, удаление профиля.
- получение объекта сущности, списка объектов сущности и связей объекта с другими объектами, создание, удаление и редактирование объекта сущности.
- создание и удаление связи объекта.
- получение учетной записи и ее проверка на устройстве.
- получение списка агентов и групп агентов.

Авторизация и получение токена

Позволяет осуществлять авторизацию и получение токена через мутацию `login`.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 53. Параметры мутации `login`

Параметр	Тип данных	Обязателен	Описание
`input →`	Объект	Да	—
`→ password`	Строка	Да	Пароль
`→ username`	Строка	Да	Логин

В ответ на мутацию возвращаются следующие данные:

- `accessToken`: Access токен. Передается в запросах как параметр авторизации типа bearer token.
- `passwordExpiredAt`: Дата истечения пароля учетной записи.
- `refreshToken`: Refresh токен. Передается при истечении срока действия access токена.

Пример 1251. Пример мутации `login`

```
mutation Login ($input: LoginRequest!) {
    login (input: $input) {
        accessToken
        passwordExpiredAt
        refreshToken
    }
}
```

Пример 1252. Пример использования переменных

```
{
    "input": {
        "password": "admin",
        "username": "admin"
    }
}
```

Список пространств

Позволяет получить список пространств в системе через запрос `listSpaces`.

URL: `{{url}}/api/evo/vm/bff/graphql`

В ответ на запрос возвращаются следующие данные:

- `spaces`: Параметры пространства

  Пример 1253. Структура данных

  ```
  spaces {
              createdAt
              createdBy
              deletedAt
              deletedBy
  ```

```
            description
            host
            id
            name
            port
            updatedAt
            updatedBy
        }
```

Пример 1254. Пример запроса `listSpaces`

```
query ListSpaces {
    listSpaces {
        spaces {
            createdAt
            createdBy
            deletedAt
            deletedBy
            description
            host
            id
            name
            port
            updatedAt
            updatedBy
        }
    }
}
```

Список тенантов

Позволяет получить список тенантов в системе через запрос `listTenants`.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 54. Параметры запроса `listTenants`

Параметр	Тип данных	Обязателен	Описание
`input →`	Объект	Нет	—
`→ filter →`	Объект	Нет	—
`→ → query`	Строка	Нет	—
`→ → statuses`	Массив [Строка]	Нет	Статус тенанта: • DISABLED — Неактивен • ENABLED — Активен

Параметр	Тип данных	Обязателен	Описание
→ orderBy	Строка	Нет	Порядок сортировки. Пример: name DESC
pageRequest →	Объект	Нет	—
→ → countTotal	Булево	Нет	Подсчет количества объектов на странице
→ → limit	Целое число	Нет	Ограничение на количество возвращаемых результатов
→ → offset	Целое число	Нет	Смещение

В ответ на запрос возвращаются следующие данные:

- tenants
- createdBy: UUID пользователя, который создал тенант.
- id: UUID тенанта.
- name: Название тенанта.
- parentId: UUID родительского тенанта.
- description: Описание тенанта.
- recordDatetime: Время создания, удаления, обновления тенанта.

```
recordDatetime {
      createdAt
      deletedAt
      updatedAt
}
```

- status: Статус тенанта (DISABLED, ENABLED).
- updatedBy: UUID пользователя, который редактировал тенант.
- creator: Информация о пользователе, который создал тенант.

```
creator {
     fullname
     id
     username
}
```

- updater: Информация о пользователе, который редактировал тенант.

```
updater {
     fullname
     id
     username
}
```

Пример 1255. Пример запроса `listTenants`

```
query listTenants ($input: ListTenantsRequest) {
    listTenants (input: $input) {
        tenants {
            createdBy
            id
            name
            parentId
            status
            updatedBy
            description
            recordDatetime {
                createdAt
                deletedAt
                updatedAt
            }
            creator {
                fullname
                username
                id
            }
            updater {
                fullname
                id
                username
            }
        }
    }
}
```

Пример 1256. Пример использования переменных

```
{
    "input": {
        "orderBy": "name asc",
        "filter": null
    }
}
```

Пример 1257. Пример ответа

```
{
```

```
    "data": {
        "listTenants": {
            "tenants": [
                {
                    "createdBy": "4a84e42f-59d3-4736-9e55-41fc0e730dd2",
                    "id": "ea2e5b16-800a-4e1c-80b3-eb4aefefd4bb",
                    "name": "1",
                    "parentId": "00000000-0000-0000-0000-000000000000",
                    "status": "ENABLED",
                    "updatedBy": null,
                    "description": "",
                    "recordDatetime": {
                        "createdAt": "2024-12-28T08:33:03.249Z",
                        "deletedAt": null,
                        "updatedAt": "2024-12-28T08:33:03.249Z"
                    },
                    "creator": {
                        "fullname": null,
                        "username": "admin",
                        "id": "4a84e42f-59d3-4736-9e55-41fc0e730dd2"
                    },
                    "updater": null
                },
                {
                    "createdBy": "4a84e42f-59d3-4736-9e55-41fc0e730dd2",
                    "id": "38c03a96-d40b-4bfa-bdda-89c6938bedaa",
                    "name": "2",
                    "parentId": "00000000-0000-0000-0000-000000000000",
                    "status": "ENABLED",
                    "updatedBy": null,
                    "description": "",
                    "recordDatetime": {
                        "createdAt": "2024-12-28T08:33:06.382Z",
                        "deletedAt": null,
                        "updatedAt": "2024-12-28T08:33:06.382Z"
                    },
                    "creator": {
                        "fullname": null,
                        "username": "admin",
                        "id": "4a84e42f-59d3-4736-9e55-41fc0e730dd2"
                    },
                    "updater": null
                }
            }
        }
    }
}
```

Список сканеров

Позволяет получить список сканеров через запрос `listScanners`.

URL: `{{url}}/api/evo/vm/bff/graphql`

Таблица 55. Параметры запроса `listScanners`

Параметр	Тип данных	Обязателен	Описание
`input →`	Объект	Нет	—
`→ filter →`	Объект	Нет	Параметры фильтра
`→ → name`	Строка	Нет	Название задачи
`→ → query`	Строка	Нет	Сквозной поиск по полям: • `name`; • `description`.
`→ pagination →`	Объект	Нет	Параметры пагинации
`→ → limit`	Целое число	Нет	Ограничение на количество возвращаемых результатов
`→ → offset`	Целое число	Нет	Смещение
`→ sorting →`	Объект	Нет	Параметры сортировки
`→ → order`	Строка	Нет	Порядок
`→ → orderBy`	Строка	Нет	Поле сортировки

В ответ на запрос возвращаются следующие данные:

- `scanners`: Параметры сканера

  Пример 1258. Структура данных

```
scanners {
            cpu
            createdAt
            createdBy
            deletedAt
            deletedBy
            description
            id
            isActive
            name
            ram
            spaceId
            spaceName
            type
```

```
            updatedAt
            updatedBy
            agentConfiguration {
                name
                ram
                workersCount
            }
        }
```

Пример 1259. Пример запроса `listScanners`

```
query ListScanners ($input: ListScannersRequest!) {
    listScanners(input: $input) {
        scanners {
            cpu
            createdAt
            createdBy
            deletedAt
            deletedBy
            description
            id
            isActive
            name
            ram
            spaceId
            spaceName
            updatedAt
            updatedBy
            agentConfiguration {
                name
                ram
                workersCount
            }
        }
    }
}
```

Пример 1260. Пример использования переменных

```
{
    "input": {
        "filter": {}
    }
}
```

Пример 1261. Пример ответа

```
{
    "data": {
```

```
        "listScanners": {
            "scanners": [
                {
                    "cpu": 1,
                    "createdAt": "2024-12-18T14:08:09.332Z",
                    "createdBy": "00000000-0000-0000-0000-000000000000",
                    "deletedAt": null,
                    "deletedBy": "admin",
                    "description": "",
                    "id": "workload-2",
                    "isActive": false,
                    "name": "workload-2",
                    "ram": 1,
                    "spaceId": "space-1",
                    "spaceName": "space-1",
                    "updatedAt": "2024-12-18T14:08:09.332Z",
                    "updatedBy": "00000000-0000-0000-0000-000000000000",
                    "agentConfiguration": [
                        {
                            "name": "vm-discovery-scan-agent",
                            "ram": 1,
                            "workersCount": 1
                        },
                        {
                            "name": "vm-discovery-agent",
                            "ram": 1,
                            "workersCount": 1
                        },
                        {
                            "name": "vm-pentest-agent",
                            "ram": 1,
                            "workersCount": 1
                        },
                        {
                            "name": "vm-data-collection-agent",
                            "ram": 1,
                            "workersCount": 1
                        }
                    ]
                }
            ]
        }
    }
}
```


Получение задачи сканирования

Позволяет получить задачу сканирования через запрос `getTask`.

URL: `{{url}}/api/evo/vm/bff/graphql`

Таблица 56. Параметры запроса `getTask`

Параметр	Тип данных	Обязателен	Описание
`id`	Строка	Да	UUID задачи сканирования
`tenantId`	Строка	Нет	UUID тенанта

В ответ на запрос возвращаются следующие данные:

- `accountIds`: UUID учетных записей для сканирования.
- `agentScanGroups`: Массив со списком UUID агентов

  Тип сканирования: Агентское.

- `bruteforceServiceList → services`: Массив со списком сервисов для подбора паролей

  Тип профиля: Тестирование на проникновение.

- `createdAt`: Дата создания задачи.
- `dateFinish`: Дата и время окончания срока действия задачи (в формате Unix).
- `dateNextLaunch`: Дата следующего запуска.
- `dateNextPaused`: Дата следующей приостановки.
- `dateOffset`: Сдвиг UTC (если указывается срок действия задачи).
- `dateStart`: Дата и время начала срока действия задачи (в формате Unix).
- `description`: Описание задачи.
- `id`: UUID задачи сканирования.
- `loggingMode`: Уровень журналирования:
  - LOGGING_MODE_DEBUG – расширенный.
  - LOGGING_MODE_INFO – информационный.
- `name`: Название задачи сканирования.
- `pausedDuration`: Длительность приостановки (в секундах).
- `pausedOffsetStart`: Время начала приостановки (в секундах).
- `pausedUnit`: Дни месяца/недели, в которые нужно приостанавливать сканирование, в формате чисел.

  Пример 1262. Структура данных

  ```
  "pausedUnit": "1,2" // понедельник, вторник
  ```

- `profileId`: UUID профиля.
- `repeatInterval`: Повторять запуск задачи каждый месяц/неделю/день.
- `repeatUnit`: Дни месяца/недели, в которые нужно запускать задачу.

Пример 1263. Структура данных

```
"repeatUnit": "3,11,19,20"
```

- `scanMode`: Тип сканирования:
  - SCAN_MODE_AGENT — Агентское сканирование.
  - SCAN_MODE_AGENTLESS — Безагентское сканирование.
- `spaceId`: ID пространства.
- `targets`: IP целей сканирования.
- `targetsExcludes`: IP исключений сканирования.
- `taskPaused`: Приостановка сканирования.
- `taskRunner`: Режим запуска:
  - TASK_RUNNER_DAY — ежедневно.
  - TASK_RUNNER_MANUAL — вручную.
  - TASK_RUNNER_MONTH — ежемесячно.
  - TASK_RUNNER_ONCE — однократно.
  - TASK_RUNNER_WEEK — еженедельно.
- `taskStatus`: Статус задачи:
  - TASK_STATUS_CANCELED — отменено.
  - TASK_STATUS_FINISHED — выполнено.
  - TASK_STATUS_INIT — не выполнялось.
  - TASK_STATUS_INTERRUPTED — ошибка.
  - TASK_STATUS_PAUSED — приостановлено.
  - TASK_STATUS_PROCESSING — выполняется.
- `timeStart`: Время запуска задачи по расписанию (в формате Unix).
- `tenantId`: UUID тенанта.
- `workloadId`: ID сканера.

Пример 1264. Пример запроса `getTask`

```
query getTask ($id: String!) {
    getTask (id: $id) {
        id
        accountIds
        agentScanGroups
        bruteforceServiceList {
            services
        }
        createdAt
        dateFinish
        dateNextLaunch
```

```
        dateNextPaused
        dateOffset
        dateStart
        description
        id
        loggingMode
        name
        pausedDuration
        pausedInterval
        pausedOffsetStart
        pausedUnit
        profileId
        repeatInterval
        repeatUnit
        scanMode
        spaceId
        targets
        targetsExcludes
        taskPaused
        taskRunner
        taskStatus
        tenantId
        timeStart
        workloadId
    }
}
```

Пример 1265. Пример использования переменных

```
{
  "id": "028924cd-19cd-43f2-8bb8-75697d269920"
}
```

Получение списка задач

Запрос `findTasks` позволяет получить список задач сканирования.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 57. Параметры запроса `findTasks`

Параметр	Тип данных	Обязателен	Описание
`filter →`	Объект	Нет	Параметры фильтра
`→ name`	Строка	Нет	Название задачи
`→ query`	Строка	Нет	Сквозной поиск по полям: • `name`; • `description`.

Параметр	Тип данных	Обязателен	Описание
→ status	Строка	Нет	Статус задачи: • TASK_STATUS_CANCELED — отменено • TASK_STATUS_FINISHED — выполнено • TASK_STATUS_INIT — не выполнялось • TASK_STATUS_INTERRUPTED — ошибка • TASK_STATUS_PAUSED — приостановлено • TASK_STATUS_PROCESSING — выполняется
→ profile	Строка	Нет	Тип профиля: • TYPE_OF_PROFILE_BASIC — Базовое сканирование • TYPE_OF_PROFILE_COMPLIANCE — Проверка соответствия • TYPE_OF_PROFILE_DISCOVERY — Обнаружение • TYPE_OF_PROFILE_PENTEST — Тестирование на проникновение • TYPE_OF_PROFILE_VULNER — Базовое сканирование
→ scanMode	Строка	Нет	Тип сканирования: • SCAN_MODE_AGENT — Агентское сканирование • SCAN_MODE_AGENTLESS — Безагентское сканирование
→ spaceId	Строка	Нет	ID пространства
→ workloadId	Строка	Нет	ID сканера
→ tenantId	Строка	Нет	UUID тенанта
pagination →	Объект	Нет	Параметры пагинации
→ limit	Целое число	Нет	Ограничение на количество возвращаемых результатов
→ offset	Целое число	Нет	Смещение
sorting →	Объект	Нет	Параметры сортировки
→ order	Строка	Нет	Порядок
→ orderBy	Строка	Нет	Поле сортировки

В ответ на запрос возвращаются следующие данные:

- `tasks`: Параметры задачи сканирования.

  Пример 1266. Структура данных

```
tasks {
    id
    accountIds
    agentScanGroups
    bruteforceServiceList {
        services
    }
    createdAt
    dateFinish
    dateNextLaunch
    dateNextPaused
    dateOffset
    dateStart
    description
    id
    loggingMode
    name
    pausedDuration
    pausedInterval
    pausedOffsetStart
    pausedUnit
    profileId
    repeatInterval
    repeatUnit
    scanMode
    spaceId
    targets
    targetsExcludes
    taskPaused
    taskRunner
    taskStatus
    tenantId
    timeStart
    workloadId
}
```

- `pageInfo`: Параметры страницы.

  Пример 1267. Структура данных

```
pageInfo {
    totalCount
    pagination {
        limit
        offset
```

```
    }
}
```

Пример 1268. Пример запроса `findTasks`

```
query findTasks ($filter: TasksFilter, $pagination: PaginationInput, $sorting:
TasksSorting) {
    findTasks (filter: $filter, pagination: $pagination, sorting: $sorting){
        tasks {
            id
           accountIds
            agentScanGroups
            bruteforceServiceList {
                services
            }
            createdAt
            dateFinish
            dateNextLaunch
            dateNextPaused
            dateOffset
            dateStart
            description
            id
            loggingMode
            name
            pausedDuration
            pausedInterval
            pausedOffsetStart
            pausedUnit
            profileId
            repeatInterval
            repeatUnit
            scanMode
            spaceId
            targets
            targetsExcludes
            taskPaused
            taskRunner
            taskStatus
            tenantId
            timeStart
            workloadId
                }
            }
}
```

Пример 1269. Пример использования переменных

```
{
    "filter":
```

```
    {
        "name": "задача"
    }
}
```

Получение истории запусков задачи сканирования

Запрос `FindTaskHistory` позволяет получить историю запусков задачи сканирования.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 58. Параметры запроса `FindTaskHistory`

Параметр	Тип данных	Обязателен	Описание
`filter →`	Объект	Нет	Фильтр по статусу сканирования
`→ status`	Строка	Нет	Статус сканирования
`taskId`	Строка	Да	UUID задачи

В ответ на запрос возвращаются следующие данные:

- `scans`: Параметры запуска сканирования.

  Пример 1270. Структура данных

  ```
  scans {
      discovered
      finished
      finishedAt
      id
      interrupted
      onlyDiscoveredHost
      scanDuration
      scanSuccess
      startedAt
      status
      targets
  }
  ```

- `taskId`: UUID задачи.

Пример 1271. Пример запроса `FindTaskHistory`

```
query FindTaskHistory {
    findTaskHistory(taskId: "8fb5ad26-4e54-432c-a320-0b869b5e59a9") {
        taskId
        scans {
            discovered
            finished
```

```
            finishedAt
            id
            interrupted
            onlyDiscoveredHost
            scanDuration
            scanSuccess
            startedAt
            status
            targets
        }
    }
}
```

Пример 1272. Пример использования переменных

```
{
  "taskId": "df846bc0-bded-42bd-9988-112dd5b9055f"
}
```

Создание задачи сканирования

Позволяет создать задачу сканирования через мутацию `createTask`.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 59. Параметры мутации `createTask`

Параметр	Тип данных	Обязателен	Описание
input →	Объект	Да	—
→ accountIds	Строка	Да, если выбран тип сканирования **Безагентское**	UUID учетных записей. Если их несколько, указываются через запятую: Пример: `"accountIds": "172b427e-ac88-4af0-847f-c30ef0cf609d,74e74f4a-4d06-4cad-8ec9-0f8410f5c578,842e047c-524d-4e20-9b02-e9cf3f0898d8"`

Параметр	Тип данных	Обязателен	Описание
→ agentScanGroups	Массив [Строка]	Да, если выбран тип сканирования **Агентское**	Массив со списком UUID агентов Пример: `"agentScanGroups": [ "bceabf60-5f56-471e-9c12-5285ff8f6b42", "603f3ba5-e2cc-4fe5-8ba3-8006d15785a2", "c2038992-3c28-458d-aa4c-037b54eccc54" ]` Тип сканирования: Агентское
→ bruteforceServiceList → services	Массив [Строка]	Да, если выбран тип профиля **Тестирование на проникновение**	Массив со списком сервисов для подбора паролей Пример: `"services": ["TYPE_OF_BRUTEFORCE_SERVICE_FTP", "TYPE_OF_BRUTEFORCE_SERVICE_FTPS" ]` Тип профиля: Тестирование на проникновение
→ dateFinish	Целое число	Нет	Дата и время окончания срока действия задачи (в формате Unix)
→ dateOffset	Целое число	Да, если заполняется срок действия задачи	Сдвиг UTC (если указывается срок действия задачи)
→ dateStart	Целое число	Нет	Дата и время начала срока действия задачи (в формате Unix)
→ description	Строка	Да	Описание задачи сканирования
→ loggingMode	Строка	Да	Уровень журналирования: • LOGGING_MODE_DEBUG — расширенный • LOGGING_MODE_INFO — информационный

Параметр	Тип данных	Обязателен	Описание
→ name	Строка	Да	Название задачи сканирования
→ pausedDuration	Целое число	Да, если используется приостановка	Длительность приостановки (в секундах)
→ pausedOffsetStart	Целое число	Да, если используется приостановка	Время начала приостановки (в секундах)
→ pausedUnit	Строка	Да, если используется приостановка	Дни месяца/недели, в которые нужно приостанавливать сканирование в формате чисел Пример: `"pausedUnit": "1,2" // понедельник, вторник`
→ profileId	Строка	Да	UUID профиля
→ repeatInterval	Целое число	Да, если задан ежемесячный, еженедельный или ежедневный запуск	Повторять запуск задачи каждый __ месяц/неделю/день
→ repeatUnit	Строка	Да, если задан ежемесячный или еженедельный запуск	Дни месяца/недели, в которые нужно запускать задачу Пример: `"repeatUnit": "3,11,19,20"`
→ scanMode	Строка	Да	Тип сканирования: • SCAN_MODE_AGENT — Агентское сканирование • SCAN_MODE_AGENTLESS — Безагентское сканирование
→ spaceId	Строка	Да	ID пространства
→ tenantId	Строка	Да	UUID тенанта

Параметр	Тип данных	Обязателен	Описание
→ targets	Строка	Да	IP целей сканирования
→ targetsExcludes	Строка	Да	IP исключений сканирования
→ taskPaused	Строка	Да	Значение в соответствии со значением переменной `taskRunner`
→ taskRunner	Строка	Да	Режим запуска: • TASK_RUNNER_DAY — ежедневно • TASK_RUNNER_MANUAL — вручную • TASK_RUNNER_MONTH — ежемесячно • TASK_RUNNER_ONCE — однократно • TASK_RUNNER_WEEK — еженедельно Приостановка невозможна при режиме запуска вручную.
→ timeStart	Целое число	Нет	Время запуска задачи по расписанию (в формате Unix)
→ workloadId	Строка	Да	ID сканера

В ответ на мутацию возвращаются следующие данные:

- `error`: Текст ошибки запроса.
- `success`: Статус запроса.
- `task`: Параметры задачи.

Пример 1273. Структура данных

```
task {
    id
    accountIds
    agentScanGroups
    bruteforceServiceList {
        services
    }
    createdAt
    dateFinish
    dateNextLaunch
    dateNextPaused
    dateOffset
    dateStart
    description
    id
```

```
    loggingMode
    name
    pausedDuration
    pausedInterval
    pausedOffsetStart
    pausedUnit
    profileId
    repeatInterval
    repeatUnit
    scanMode
    spaceId
    targets
    targetsExcludes
    taskPaused
    taskRunner
    taskStatus
    tenantId
    timeStart
    workloadId
}
```

Пример 1274. Пример мутации `createTask`

```
mutation createTask ($input: CreateTaskInput!) {
    createTask (input: $input) {
        error
        success
        task {
            id
        }
        tenantId
    }
}
```

Пример 1275. Пример использования переменных

```
{
  "input": {
    "accountIds": "7ef2452e-167a-45d3-9957-815e7365f6df",
    "agentScanGroups": [],
    "bruteforceServiceList": {
      "services": []
    },
    "dateFinish": 0,
    "dateOffset": 0,
    "dateStart": 0,
    "description": "",
    "loggingMode": "LOGGING_MODE_INFO",
    "name": "задача api 16",
    "organizationId": "",
```

```
    "pausedDuration": 0,
    "pausedInterval": 0,
    "pausedOffsetStart": 0,
    "pausedUnit": "",
    "profileId": "458f4e3a-0308-4280-bd1b-6f8182bad54c",
    "repeatInterval": 0,
    "repeatUnit": "",
    "spaceId": "space-0",
    "scanMode": "SCAN_MODE_AGENTLESS",
    "targets": "10.99.101.50",
    "targetsExcludes": "",
    "taskPaused": "TASK_RUNNER_MANUAL",
    "taskRunner": "TASK_RUNNER_MANUAL",
    "taskStatus": "TASK_STATUS_INIT",
    "tenantId": "f88b30ff-7862-489a-a6dd-c82c23e5ea21",
    "timeStart": 0,
    "workloadId": "workload-2"
  }
}
```


Пример 1276. Пример ответа


```
{
    "data": {
        "createTask": {
            "error": null,
            "success": true,
            "task": {
                "id": "ed615552-ad0d-460c-86e5-68c7096fda86"
            }
            "tenantId": "f88b30ff-7862-489a-a6dd-c82c23e5ea21"
        }
    }
}
```


Редактирование задачи сканирования

Позволяет редактировать задачу сканирования через мутацию `updateTask`.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 60. Параметры мутации `updateTask`

Параметр	Тип данных	Обязателен	Описание
update →	Объект	Да	Параметры задачи

Параметр	Тип данных	Обязателен	Описание
→ accountIds	Строка	Да	UUID учетных записей. Если их несколько, указываются через запятую: Пример: `"accountIds": "172b427e-ac88-4af0-847f-c30ef0cf609d,74e74f4a-4d06-4cad-8ec9-0f8410f5c578,842e047c-524d-4e20-9b02-e9cf3f0898d8"`
→ agentScanGroups	Массив [Строка]	Да	Массив со списком UUID агентов Пример: `"agentScanGroups": [ "bceabf60-5f56-471e-9c12-5285ff8f6b42", "603f3ba5-e2cc-4fe5-8ba3-8006d15785a2", "c2038992-3c28-458d-aa4c-037b54eccc54" ]` Тип сканирования: Агентское
→ bruteforceServiceList → services	Массив [Строка]	Нет	Массив со списком сервисов для подбора паролей Пример: `"services": ["TYPE_OF_BRUTEFORCE_SERVICE_FTP", "TYPE_OF_BRUTEFORCE_SERVICE_FTPS" ]` Тип профиля: Тестирование на проникновение
→ dateFinish	Целое число	Нет	Дата и время окончания срока действия задачи (в формате Unix)
→ dateOffset	Целое число	Нет	Сдвиг UTC (если указывается срок действия задачи)
→ dateStart	Целое число	Нет	Дата и время начала срока действия задачи (в формате Unix)

Параметр	Тип данных	Обязателен	Описание
→ description	Строка	Нет	Описание задачи сканирования
→ id	Строка	Да	UUID задачи
→ loggingMode	Строка	Да	Уровень журналирования: • LOGGING_MODE_DEBUG — расширенный • LOGGING_MODE_INFO — информационный
→ name	Строка	Нет	Название задачи сканирования
→ pausedDuration	Целое число	Нет	Длительность приостановки (в секундах)
→ pausedOffsetStart	Целое число	Нет	Время начала приостановки (в секундах)
→ pausedUnit	Строка	Нет	Дни месяца/недели, в которые нужно приостанавливать сканирование в формате чисел Пример: `"pausedUnit": "1,2" // понедельник, вторник`
→ profileId	Строка	Да	UUID профиля
→ repeatInterval	Целое число	Нет	Повторять запуск задачи каждый __ месяц/неделю/день
→ repeatUnit	Строка	Нет	Дни месяца/недели, в которые нужно запускать задачу Пример: `"repeatUnit": "3,11,19,20"`
→ scanMode	Строка	Да	Тип сканирования: • SCAN_MODE_AGENT — Агентское сканирование • SCAN_MODE_AGENTLESS — Безагентское сканирование
→ spaceId	Строка	Да	ID пространства
→ targets	Строка	Да	IP целей сканирования

Параметр	Тип данных	Обязателен	Описание
→ targetsExcludes	Строка	Да	IP исключений сканирования
→ taskPaused	Строка	Да	Значение в соответствии со значением переменной taskRunner
→ taskRunner	Строка	Да	Режим запуска: • TASK_RUNNER_DAY — ежедневно • TASK_RUNNER_MANUAL — вручную • TASK_RUNNER_MONTH — ежемесячно • TASK_RUNNER_ONCE — однократно • TASK_RUNNER_WEEK — еженедельно Приостановка невозможна при режиме запуска вручную.
→ taskStatus	Строка	Нет	—
→ timeStart	Целое число	Нет	Время запуска задачи по расписанию (в формате Unix)
→ workloadId	Строка	Да	ID сканера

В ответ на мутацию возвращаются следующие данные:

- `error`: Текст ошибки запроса.
- `success`: Статус запроса.
- `task`: Параметры задачи.

Пример 1277. Структура данных

```
task {
    id
    accountIds
    agentScanGroups
    bruteforceServiceList {
        services
    }
    createdAt
    dateFinish
    dateNextLaunch
    dateNextPaused
    dateOffset
    dateStart
    description
    id
```

```
    loggingMode
    name
    pausedDuration
    pausedInterval
    pausedOffsetStart
    pausedUnit
    profileId
    repeatInterval
    repeatUnit
    scanMode
    spaceId
    targets
    targetsExcludes
    taskPaused
    taskRunner
    taskStatus
    timeStart
    workloadId
}
```

Удаление задачи сканирования

Позволяет удалить задачу сканирования через мутацию `deleteTask`.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 61. Параметры мутации `deleteTask`

Параметр	Тип данных	Обязателен	Описание
delete →	Объект	Да	—
→ id	Строка	Да	UUID профиля

В ответ на мутацию возвращаются следующие данные:

- `error`: Текст ошибки запроса.
- `success`: Статус запроса.
- `task`: Параметры задачи.

  Пример 1278. Структура данных

  ```
  task {
      id
  }
  ```

Пример 1279. Пример мутации `deleteTask`

```
mutation deleteTask ($delete: DeleteTaskInput!) {
    deleteTask (delete: $delete) {
        error
```

```
        success
    }
}
```

Пример 1280. Пример использования переменных

```
{
  "delete": {
    "id": "df846bc0-bded-42bd-9988-112dd5b9055f"
  }
}
```

Запуск задачи сканирования

Позволяет осуществлять запуск задачи сканирования через мутацию `runTask`.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 62. Параметры мутации `runTask`

Параметр	Тип данных	Обязателен	Описание
run →	Объект	Да	—
→ id	Строка	Да	UUID задачи

В ответ на мутацию возвращаются следующие данные:

- `error`: Текст ошибки запроса.
- `success`: Статус запроса.
- `task`: UUID задачи.

  Пример 1281. Структура данных

  ```
  tasks {
      id
  }
  ```

Пример 1282. Пример мутации `runTask`

```
mutation runTask ($run: RunTaskInput!) {
    runTask (run: $run) {
        error
        success
    }
}
```

Пример 1283. Пример использования переменных

```
{
```

```
  "run": {
    "id": "df846bc0-bded-42bd-9988-112dd5b9055f"
  }
}
```

Отмена задачи сканирования

Позволяет отменить запущенную задачу сканирования через мутацию `cancelTask`.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 63. Параметры мутации `cancelTask`

Параметр	Тип данных	Обязателен	Описание
cancel →	Объект	Да	Параметры задачи
→ id	Строка	Да	UUID задачи

В ответ на мутацию возвращаются следующие данные:

- `error`: Текст ошибки запроса.
- `success`: Статус запроса.
- `task`: Параметры задачи.

  Пример 1284. Структура данных

  ```
  task {
      id
  }
  ```

Пример 1285. Пример мутации `cancelTask`

```
mutation cancelTask ($cancel: CancelTaskInput!) {
    cancelTask (cancel: $cancel) {
        error
        success
        task {
            id
        }
    }
}
```

Пример 1286. Пример использования переменных

```
{
  "cancel": {
    "id": "df846bc0-bded-42bd-9988-112dd5b9055f"
  }
}
```

Приостановка задачи сканирования

Позволяет приостановить задачу сканирования через мутацию `pauseTask`.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 64. Параметры мутации `pauseTask`

Параметр	Тип данных	Обязателен	Описание
pause →	Объект	Да	—
→ id	Строка	Да	UUID задачи

В ответ на мутацию возвращаются следующие данные:

- `error`: Текст ошибки запроса.
- `success`: Статус запроса.
- `task`: UUID задачи.

Пример 1287. Структура данных

```
task {
    id
}
```

Пример 1288. Пример мутации `pauseTask`

```
mutation pauseTask ($pause: PauseTaskInput!) {
    pauseTask (pause: $pause) {
        error
        success
    }
}
```

Пример 1289. Пример использования переменных

```
{
  "pause": {
    "id": "ad755637-63d1-4834-97e3-b963257af96f"
  }
}
```

Возобновление задачи

Позволяет возобновить задачу через мутацию `resumeTask`.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 65. Параметры мутации `resumeTask`

Параметр	Тип данных	Обязателен	Описание
resume →	Объект	Да	—
→ id	Строка	Да	UUID задачи

В ответ на мутацию возвращаются следующие данные:

- `error`: Текст ошибки запроса.
- `success`: Статус запроса.
- `task`: UUID задачи.

Пример 1290. Структура данных

```
task {
    id
}
```

Пример 1291. Пример мутации `resumeTask`

```
mutation resumeTask ($resume: ResumeTaskInput!) {
    resumeTask (resume: $resume) {
        error
        success
    }
}
```

Пример 1292. Пример использования переменных

```
{
  "resume": {
    "id": ""
  }
}
```

Получение профиля

Позволяет получить профиль через запрос `getProfile`.

URL: `{{url}}/api/evo/vm/bff/graphql`

Таблица 66. Параметры запроса `getProfile`

Параметр	Тип данных	Обязателен	Описание
`id`	Строка	Да	UUID профиля

В ответ на запрос возвращаются следующие данные:

- `profiles`: Параметры профиля.

- `bruteforceSettings`: Параметры сервиса подбора паролей.
- `emptyPassword`: Проверять пустые пароли.
- `endAfterFirstMatch`: Завершать проверку узла после совпадения.
- `loginAsPassword`: Проверять логин в качестве пароля.
- `threadCount`: Количество одновременно обрабатываемых логинов.
- `services`: Параметры сервисов.

Пример 1293. Структура данных

```
services {
        loginFile: String
        loginFileHash: String
        loginFileId: String
        name: String
        passwordFile: String
        passwordFileHash: String
        passwordFileId: String
        metadata {
            key: String
            value: String
        }
}
```

- `description`: Описание профиля.
- `importSettings`: Параметры импорта.

Пример 1294. Структура данных

```
importSettings {
    defineOs: Boolean
    defineVersionService: Boolean
    importOpenPort: Boolean
    importSecurityOptions: Boolean
    importSoft: Boolean
    importUsers: Boolean
    importWindowsUpdates: Boolean
}
```

- `name`: Название профиля.
- `id`: UUID профиля.
- `nmapSettings`: Параметры обнаружения.
- `authSsh`: Порт авторизации SSH

Тип профиля: Проверка стандартов.

- `connectionTimeout`: Таймаут соединения (в минутах).

Тип профиля: Проверка стандартов, Тестирование на проникновение.

- `dnsNameResolver`: Разрешение имен DNS

  Тип профиля: Проверка стандартов, Тестирование на проникновение.

- `extendSettings`: Пользовательский режим работы.

  Тип профиля: Тестирование на проникновение.

- `icmp`: ICMP пингование (обнаружение хостов).

  Тип профиля: Проверка стандартов, Тестирование на проникновение.

- `nmapString`: Параметры командной строки (если выбран Пользовательский режим работы).

  Тип профиля: Тестирование на проникновение.

- `scanTcp`: Сканирование портов TCP.

  Тип профиля: Проверка стандартов, Тестирование на проникновение.

- `scanUdp`: Сканирование портов UDP.

  Тип профиля: Проверка стандартов, Тестирование на проникновение.

- `tcpSyn`: Порты для TCP SYN пингования (обнаружение хостов).

  Тип профиля: Проверка стандартов, Тестирование на проникновение.

- `timeTemplate`: Шаблон настроек времени:
  - NMAP_TIME_TEMPLATE_T0 – Очень медленный (T0)
  - NMAP_TIME_TEMPLATE_T1 – Медленный (T1)
  - NMAP_TIME_TEMPLATE_T2 – Умеренный (T2)
  - NMAP_TIME_TEMPLATE_T3 – Стандартный (T3)
  - NMAP_TIME_TEMPLATE_T4 – Быстрый (T4)
  - NMAP_TIME_TEMPLATE_T5 – Очень быстрый (T5)

  Тип профиля: Проверка стандартов, Тестирование на проникновение.

- `udp`: Порты для UDP пингования (обнаружение хостов).

  Тип профиля: Проверка стандартов, Тестирование на проникновение.

- `nseSettings`: Параметры безопасных и небезопасных проверок.
- `categories`: Использовать:
  - safe – безопасные проверки.
  - unsafe – небезопасные проверки.

  Тип профиля: Тестирование на проникновение.

- `pentestSettings`: Параметры тестирования на проникновение.
- `extendSettings`: Пользовательские настройки.
- `mode`: Тип сканирования портов:
  - PENTEST_MODE_DEFAULT – Базовый
  - PENTEST_MODE_CUSTOM – Пользовательский список
  - PENTEST_MODE_FULL – Все порты
  - PENTEST_MODE_TOP25 – Топ 25
  - PENTEST_MODE_TOP100 – Топ 100
- `tcpPorts`: Порты для TCP пингования.
- `udpPorts`: Порты для UDP пингования.
- `complianceSettings`: Параметры проверок соответствия.
- `benchmarks`: Массив со списком стандартов.
- `pentestSettings`: Параметры тестирования на проникновение.

  Тип профиля: Проверка стандартов.

- `typeOfProfile`: Тип профиля:
  - TYPE_OF_PROFILE_BASIC – Базовое сканирование
  - TYPE_OF_PROFILE_COMPLIANCE – Проверка соответствия
  - TYPE_OF_PROFILE_DISCOVERY – Обнаружение
  - TYPE_OF_PROFILE_PENTEST – Тестирование на проникновение
  - TYPE_OF_PROFILE_VULNER – Базовое сканирование
- `scanMode`: Режим сканирования:
  - SCAN_MODE_AGENTLESS – Безагентское сканирование
  - SCAN_MODE_AGENT – Агентское сканирование
- `tenantId`: UUID тенанта.

Пример 1295. Пример запроса `getProfile`

```
query getProfile ($id: String!) {
    getProfile (id: $id) {
        description
        discoveryType
        id
        name
        typeOfProfile
    }
}
```

Пример 1296. Пример использования переменных

```
{
  "id": ""
}
```

Получение списка профилей

Запрос `findProfiles` позволяет получить список профилей.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 67. Параметры запроса `findProfiles`

Параметр	Тип данных	Обязателен	Описание
`filter →`	Объект	Нет	Параметры фильтра
`→ name`	Строка	Нет	Название задачи
`→ query`	Строка	Нет	Сквозной поиск по полям: • `name`; • `description`.
`→ scanMode`	Строка	Нет	Тип сканирования: • SCAN_MODE_AGENT — Агентское сканирование • SCAN_MODE_AGENTLESS — Безагентское сканирование
`→ typeOfProfile`	Строка	Нет	Тип профиля: • TYPE_OF_PROFILE_BASIC — Базовое сканирование • TYPE_OF_PROFILE_COMPLIANCE — Проверка соответствия • TYPE_OF_PROFILE_DISCOVERY — Обнаружение • TYPE_OF_PROFILE_PENTEST — Тестирование на проникновение • TYPE_OF_PROFILE_VULNER — Базовое сканирование
`pagination →`	Объект	Нет	Параметры пагинации
`→ limit`	Целое число	Нет	Ограничение на количество возвращаемых результатов
`→ offset`	Целое число	Нет	Смещение
`sorting →`	Объект	Нет	Параметры сортировки

Параметр	Тип данных	Обязателен	Описание
→ order	Строка	Нет	Порядок
→ orderBy	Строка	Нет	Поле сортировки: • PROFILE_ORDER_BY_DESCRIPTION — О писание профиля • PROFILE_ORDER_BY_NAME — Название профиля • PROFILE_ORDER_BY_TYPE — Тип профиля

В ответ на запрос возвращаются следующие данные:

- profiles: Параметры профиля.

  Пример 1297. Структура данных

```
profiles {
    description
    discoveryType
    id
    name
    scanMode
    typeOfProfile
    complianceSettings {
        benchmarks
    }
    nmapSettings {
        authSsh
        connectionTimeout
        dnsNameResolver
        extendSettings
        icmp
        nmapString
        scanTcp
        scanUdp
        tcpSyn
        timeTemplate
        udp
    }
    nseSettings {
        scripts
        categories
    }
    pentestSettings {
        extendSettings
        mode
        nmapString
        tcpPorts
```

```
        udpPorts
    }
    bruteforceSettings {
        emptyPassword
        endAfterFirstMatch
        loginAsPassword
        threadCount
        services {
            loginFile
            loginFileHash
            loginFileId
            name
            passwordFile
            passwordFileHash
            passwordFileId
            metadata {
                key
                value
            }
        }
    }
    importSettings {
        defineOs
        defineVersionService
        importOpenPort
        importSecurityOptions
        importSoft
        importUsers
        importWindowsUpdates
    }
}
```

- `pageInfo`: Параметры страницы.

  Пример 1298. Структура данных

```
pageInfo {
    totalCount
    pagination {
        limit
        offset
    }
}
```

Пример 1299. Пример запроса `findProfiles`

```
{
    "filter":
    {
        "name": "профиль"
```

```
    }
}
```

Пример 1300. Пример использования переменных

```
query findProfiles ($filter: ProfilesFilter, $pagination: PaginationInput, $sorting:
ProfilesSorting) {
    findProfiles (filter: $filter, pagination: $pagination, sorting: $sorting) {
        profiles {
            description
            discoveryType
            id
            name
            scanMode
            typeOfProfile
            tenantId
            complianceSettings {
                benchmarks
            }
            nmapSettings {
                authSsh
                connectionTimeout
                dnsNameResolver
                extendSettings
                icmp
                nmapString
                scanTcp
                scanUdp
                tcpSyn
                timeTemplate
                udp
            }
            nseSettings {
                scripts
                categories
            }
            pentestSettings {
                extendSettings
                mode
                nmapString
                tcpPorts
                udpPorts
            }
            bruteforceSettings {
                emptyPassword
                endAfterFirstMatch
                loginAsPassword
                threadCount
                services {
                    loginFile
```

```
                    loginFileHash
                    loginFileId
                    name
                    passwordFile
                    passwordFileHash
                    passwordFileId
                    metadata {
                        key
                        value
                    }
                }
            }
            importSettings {
                defineOs
                defineVersionService
                importOpenPort
                importSecurityOptions
                importSoft
                importUsers
                importWindowsUpdates
            }
        }
        pageInfo {
            totalCount
        }
    }
}
```

Пример 1301. Пример ответа

```
{
    "data": {
        "findProfiles": {
            "profiles": [
                {
                    "description": "",
                    "discoveryType": "DISCOVERY_TYPE_NMAP",
                    "id": "00b42456-65bc-4096-ac30-703022c8f21c",
                    "name": "Тестовый",
                    "scanMode": "SCAN_MODE_AGENTLESS",
                    "typeOfProfile": "TYPE_OF_PROFILE_PENTEST",
                    "tenantId": "f88b30ff-7862-489a-a6dd-c82c23e5ea21",
                    "complianceSettings": {
                        "benchmarks": []
                    },
                    "nmapSettings": {
                        "authSsh": "22",
                        "connectionTimeout": 2,
                        "dnsNameResolver": false,
                        "extendSettings": false,
```

```
                        "icmp": false,
                        "nmapString": "--privileged
-PS21,22,23,25,53,80,110,123,135,137,143,161,389,443,445,515,554,631,3389,5060,5061,59
85,8080,8600,9100 -PU53,67,68,161 -sS -sU -R -p
T:22,80,111,135,139,389,443,445,515,554,631,5060,5061,5985,8600,9100,U:161 -T4 -oX -
--host-timeout 2m",
                        "scanTcp":
"22,80,135,139,389,443,445,515,554,631,5060,5061,8600,9100",
                        "scanUdp": "161",
                        "tcpSyn":
"21,22,23,25,53,80,110,123,135,137,143,161,389,443,445,515,554,631,3389,5060,5061,8080
,8600,9100",
                        "timeTemplate": "NMAP_TIME_TEMPLATE_T4",
                        "udp": "53,67,68,161"
                    },
                    "nseSettings": {
                        "scripts": [],
                        "categories": [
                            "safe"
                        ]
                    },
                    "pentestSettings": {
                        "extendSettings": false,
                        "mode": "PENTEST_MODE_DEFAULT",
                        "nmapString": "--privileged
-PS21,22,23,25,53,80,110,123,135,137,143,161,389,443,445,515,554,631,3389,5060,5061,59
85,8080,8600,9100 -PU53,67,68,161 -sS -sU -R -p
T:22,80,111,135,139,389,443,445,515,554,631,5060,5061,5985,8600,9100,U:161 -T4 -oX -
--host-timeout 2m",
                        "tcpPorts": "",
                        "udpPorts": ""
                    },
                    "bruteforceSettings": {
                        "emptyPassword": false,
                        "endAfterFirstMatch": false,
                        "loginAsPassword": false,
                        "threadCount": 1,
                        "services": [
                            {
                                "loginFile": "",
                                "loginFileHash": "",
                                "loginFileId": "",
                                "name": "TYPE_OF_BRUTEFORCE_SERVICE_FTP",
                                "passwordFile": "",
                                "passwordFileHash": "",
                                "passwordFileId": "",
                                "metadata": [
                                    {
                                        "key": "target-port",
                                        "value": "21"
                                    }
```

```
                            ]
                        },
                        {
                            "loginFile": "",
                            "loginFileHash": "",
                            "loginFileId": "",
                            "name": "TYPE_OF_BRUTEFORCE_SERVICE_FTPS",
                            "passwordFile": "",
                            "passwordFileHash": "",
                            "passwordFileId": "",
                            "metadata": [
                                {
                                    "key": "target-port",
                                    "value": "21"
                                }
                            ]
                        },
                        {
                            "loginFile": "",
                            "loginFileHash": "",
                            "loginFileId": "",
                            "name": "TYPE_OF_BRUTEFORCE_SERVICE_HTTPS_GET",
                            "passwordFile": "",
                            "passwordFileHash": "",
                            "passwordFileId": "",
                            "metadata": [
                                {
                                    "key": "target-port",
                                    "value": "443"
                                }
                            ]
                        }
                    ]
                },
                "importSettings": {
                    "defineOs": false,
                    "defineVersionService": false,
                    "importOpenPort": false,
                    "importSecurityOptions": false,
                    "importSoft": false,
                    "importUsers": false,
                    "importWindowsUpdates": false
                }
            },
...
```

Создание профиля

Позволяет создать профиль через мутацию `createProfile`.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 68. Параметры мутации `createProfile`

Параметр	Тип данных	Обязателен	Описание
input →	Объект	Да	Параметры профиля
→ description	Строка	Нет	Описание профиля
→ name	Строка	Да	Название профиля
→ typeOfProfile	Строка	Да	Тип профиля: • TYPE_OF_PROFILE_BASIC — Базовое сканирование • TYPE_OF_PROFILE_COMPLIANCE — Проверка соответствия • TYPE_OF_PROFILE_DISCOVERY — Обнаружение • TYPE_OF_PROFILE_PENTEST — Тестирование на проникновение • TYPE_OF_PROFILE_VULNER — Базовое сканирование
→ scanMode	Строка	Да	Режим сканирования: • SCAN_MODE_AGENTLESS — Безагентское сканирование • SCAN_MODE_AGENT — Агентское сканирование
→ tenantId	Строка	Да	UUID тенанта

В ответ на мутацию возвращаются следующие данные:

- `error`: Текст ошибки запроса.
- `success`: Статус запроса.

Пример 1302. Пример мутации `createProfile`

```
mutation createProfile ($input: CreateProfileInput!) {
    createProfile (input: $input) {
        error
        success
        profile {
            id
        }
    }
}
```

Пример 1303. Пример использования переменных

```
{
  "input": {
    "name": "профиль",
    "description": "",
    "typeOfProfile": "TYPE_OF_PROFILE_BASIC",
    "scanMode": "SCAN_MODE_AGENTLESS"
    "tenantId": "ea2e5b16-800a-4e1c-80b3-eb4aefefd4bb"
  }
}
```

Удаление профиля

Позволяет удалить профиль сканирования через мутацию `deleteProfile`.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 69. Параметры мутации `deleteProfile`

Параметр	Тип данных	Обязателен	Описание
delete →	Объект	Да	—
→ id	Строка	Да	UUID профиля

В ответ на мутацию возвращаются следующие данные:

- `error`: Текст ошибки запроса.
- `success`: Статус запроса.
- `profile`: UUID профиля.

  Пример 1304. Структура данных

  ```
  profile {
      id
  }
  ```

Пример 1305. Пример мутации `deleteProfile`

```
mutation deleteProfile ($delete: DeleteProfileInput!) {
    deleteProfile (delete: $delete) {
        error
        success
    }
}
```

Пример 1306. Пример использования переменных

```
{
```

```
  "delete": {
    "id": "02210d31-1cf1-4501-abc1-94484f011348"
  }
}
```

Редактирование профиля

Позволяет редактировать профиль через мутацию `updateProfile`.

URL: `{{url}}/api/evo/core/bff/graphql`

Таблица 70. Параметры мутации `updateProfile`

Параметр	Тип данных	Обязателен	Описание
update →	Объект	Да	Параметры профиля
→ id	Строка	Да	UUID профиля
→ bruteforceSettings →	Объект	Нет	Параметры сервиса подбора паролей
→ → emptyPassword	Логический	Нет	Проверять пустые пароли
→ → endAfterFirstMatch	Логический	Нет	Завершать проверку узла после совпадения
→ → loginAsPassword	Логический	Нет	Проверять логин в качестве пароля
→ → threadCount	Целое число	Нет	Количество одновременно обрабатываемых логинов

Параметр	Тип данных	Обязателен	Описание
→ → services	Объект `services { loginFile: String loginFileHash: String loginFileId: String name: String passwordFile: String passwordFileHash: String passwordFileId: String metadata { key: String value: String } }`	Нет	Параметры сервисов
→ description	Строка	Нет	Описание профиля
→ discoveryType	Строка	Да	—

Параметр	Тип данных	Обязателен	Описание
→ importSettings	Объект `importSettings { defineOs: Boolean defineVersionService: Boolean importOpenPort: Boolean importSecurityOptions: Boolean importSoft: Boolean importUsers: Boolean importWindowsUpdates: Boolean }`	Да	Параметры импорта
→ name	Строка	Нет	Название профиля
→ nmapSettings →	Объект	Да	Параметры обнаружения
→ → authSsh	Строка	Да	Порт авторизации SSH Тип профиля: Проверка стандартов
→ → connectionTimeout	Целое число	Да	Таймаут соединения (в минутах) Тип профиля: Проверка стандартов, Тестирование на проникновение
→ → dnsNameResolver	Логический	Да	Разрешение имен DNS Тип профиля: Проверка стандартов, Тестирование на проникновение
→ → extendSettings	Логический	Да	Пользовательский режим работы Тип профиля: Тестирование на проникновение
→ → icmp	Логический	Да	ICMP пингование (обнаружение хостов) Тип профиля: Проверка стандартов, Тестирование на проникновение

Параметр	Тип данных	Обязателен	Описание
→ → nmapString	Строка	Да	Параметры командной строки (если выбран Пользовательский режим работы) Тип профиля: Тестирование на проникновение
→ → scanTcp	Строка	Да	Сканирование портов TCP Тип профиля: Проверка стандартов, Тестирование на проникновение
→ → scanUdp	Строка	Да	Сканирование портов UDP Тип профиля: Проверка стандартов, Тестирование на проникновение
→ → tcpSyn	Строка	Да	Порты для TCP SYN пингования (обнаружение хостов) Тип профиля: Проверка стандартов, Тестирование на проникновение
→ → timeTemplate	Строка	Да	Шаблон настроек времени (обнаружение хостов): • NMAP_TIME_TEMPLATE_T0 — Очень медленный (T0) • NMAP_TIME_TEMPLATE_T1 — Медленный (T1) • NMAP_TIME_TEMPLATE_T2 — Умеренный (T2) • NMAP_TIME_TEMPLATE_T3 — Стандартный (T3) • NMAP_TIME_TEMPLATE_T4 — Быстрый (T4) • NMAP_TIME_TEMPLATE_T5 — Очень быстрый (T5) Тип профиля: Проверка стандартов, Тестирование на проникновение
→ → udp	Строка	Да	Порты для UDP пингования (обнаружение хостов) Тип профиля: Проверка стандартов, Тестирование на проникновение

Параметр	Тип данных	Обязателен	Описание
→ nseSettings →	Объект	Нет	Параметры безопасных и небезопасных проверок
→ → categories	Массив [Строка]	Нет	Использовать: • safe — безопасные проверки • unsafe — небезопасные проверки Тип профиля: Тестирование на проникновение
→ pentestSettings →	—	Нет	Параметры тестирования на проникновение
→ → extendSettings	Логический	Нет	Пользовательские настройки
→ → mode	Строка	Нет	Тип сканирования портов: • PENTEST_MODE_DEFAULT — Базовый • PENTEST_MODE_CUSTOM — Пользовательский список • PENTEST_MODE_FULL — Все порты • PENTEST_MODE_TOP25 — Топ 25 • PENTEST_MODE_TOP100 — Топ 100
→ → tcpPorts	Строка	Нет	Порты для TCP пингования
→ → udpPorts	Строка	Нет	Порты для UDP пингования
→ complianceSettings →	—	Нет	Параметры проверок соответствия
→ → benchmarks	Массив [Строка]	Да	Массив со списком стандартов Тип профиля: Проверка стандартов

Параметр	Тип данных	Обязателен	Описание
→ typeOfProfile	Строка	Да	Тип профиля: • TYPE_OF_PROFILE_BASIC — Базовое сканирование • TYPE_OF_PROFILE_COMPLIANCE — Проверка соответствия • TYPE_OF_PROFILE_DISCOVERY — Обнаружение • TYPE_OF_PROFILE_PENTEST — Тестирование на проникновение • TYPE_OF_PROFILE_VULNER — Базовое сканирование
→ scanMode	Строка	Да	Режим сканирования: • SCAN_MODE_AGENTLESS — Безагентское сканирование • SCAN_MODE_AGENT — Агентское сканирование

В ответ на мутацию возвращаются следующие данные:

- `error`: Текст ошибки запроса.
- `success`: Статус запроса.
- `profile`: Параметры профиля.

Пример 1307. Структура данных

```
profiles {
    description
    discoveryType
    id
    name
    typeOfProfile
    complianceSettings {
        benchmarks
    }
    nmapSettings {
        authSsh
        connectionTimeout
        dnsNameResolver
        extendSettings
        icmp
        nmapString
        scanTcp
        scanUdp
        tcpSyn
```

```
        timeTemplate
        udp
    }
    nseSettings {
        scripts
        categories
    }
    pentestSettings {
        extendSettings
        mode
        nmapString
        tcpPorts
        udpPorts
    }
    bruteforceSettings {
        emptyPassword
        endAfterFirstMatch
        loginAsPassword
        threadCount
        services {
            loginFile
            loginFileHash
            loginFileId
            name
            passwordFile
            passwordFileHash
            passwordFileId
            metadata {
                key
                value
            }
        }
    }
    importSettings {
        defineOs
        defineVersionService
        importOpenPort
        importSecurityOptions
        importSoft
        importUsers
        importWindowsUpdates
    }
}
```

Пример 1308. Пример мутации `updateProfile`

```
mutation updateProfile ($update: UpdateProfileInput!) {
    updateProfile (update: $update) {
        error
        success
```

```
    }
}
```

Пример 1309. Пример использования переменных

```
{
  "update": {
    "id": "00b42456-65bc-4096-ac30-703022c8f21c",
    "complianceSettings": { "benchmarks": [] },
    "discoveryType": "DISCOVERY_TYPE_NMAP",
    "name": "Тестовый 1",
    "importSettings": {
        "defineOs": true,
        "defineVersionService": true,
        "importOpenPort": true,
        "importSecurityOptions": true,
        "importSoft": true,
        "importUsers": true,
        "importWindowsUpdates": true
    },
    "nmapSettings": {
        "authSsh": "22",
        "connectionTimeout": 2,
        "dnsNameResolver": false,
        "extendSettings": false,
        "icmp": false,
        "nmapString": "--privileged
-PS21,22,23,25,53,80,110,123,135,137,143,161,389,443,445,515,554,631,3389,5060,5061,59
85,8080,8600,9100 -PU53,67,68,161 -sS -sU -R -p
T:22,80,111,135,139,389,443,445,515,554,631,5060,5061,5985,8600,9100,U:161 -T4 -oX -
--host-timeout 2m",
        "scanTcp": "22,80,135,139,389,443,445,515,554,631,5060,5061,8600,9100",
        "scanUdp": "161",
        "tcpSyn":
"21,22,23,25,53,80,110,123,135,137,143,161,389,443,445,515,554,631,3389,5060,5061,8080
,8600,9100",
        "timeTemplate": "NMAP_TIME_TEMPLATE_T4",
        "udp": "53,67,68,161"
    },
    "scanMode": "SCAN_MODE_AGENTLESS"
    }
}
```

Пример 1310. Пример ответа

```
{
    "data": {
        "updateProfile": {
            "error": null,
            "success": true
```

```
        }
    }
}
```

Получение списка стандартов

Позволяет получить список стандартов через запрос `FindAgentToScan`.

URL: `{{url}}/api/evo/vm/bff/graphql`

Таблица 71. Параметры запроса `FindAgentToScan`

Параметр	Тип данных	Обязателен	Описание
`pagination →`	Объект	Нет	Параметры пагинации
`→ limit`	Целое число	Нет	Ограничение на количество возвращаемых результатов
`→ offset`	Целое число	Нет	Смещение
`sorting →`	Объект	Нет	Параметры сортировки
`→ order`	Строка	Нет	Порядок
`→ orderBy`	Строка	Нет	Поле сортировки

В ответ на запрос возвращаются следующие данные:

- `benchmarks`: Параметры стандартов.
- `name`: Название стандарта.

Пример 1311. Пример запроса `FindAgentToScan`

```
query FindAgentToScan {
    findBenchmarks {
        benchmarks {
            name
        }
    }
}
```

Пример 1312. Пример ответа

```
{
    "data": {
        "findBenchmarks": {
            "benchmarks": [
                {
                    "name": "CIS Cisco ASA 9.x Firewall Benchmark v1.1.0"
                },
```

```
                {
                    "name": "CIS Cisco IOS 17.x Benchmark v2.0.0"
                },
                {
                    "name": "CIS Cisco IOS XE 17.x Benchmark v2.1.1"
                },
                {
                    "name": "CIS Cisco IOS XR 7.x v1.0.0"
                },
                {
                    "name": "CIS Cisco NX-OS Benchmark v1.1.0"
                },
                {
                    "name": "CIS Microsoft Windows 10 Enterprise Benchmark v3.0.0"
                },
                {
                    "name": "CIS Microsoft Windows 10 Stand-alone Benchmark v3.0.0"
                },
                {
                    "name": "CIS Microsoft Windows 11 Enterprise Benchmark v3.0.0"
                },
                {
                    "name": "CIS Microsoft Windows 11 Stand-alone Benchmark v3.0.0"
                },
                {
                    "name": "CIS Microsoft Windows Server 2016 Benchmark v3.0.0"
                },
                {
                    "name": "CIS Microsoft Windows Server 2022 Benchmark v3.0.0"
                },
                {
                    "name": "CIS Red Hat Enterprise Linux 8 Benchmark v3.0.0"
                },
                {
                    "name": "CIS Red Hat Enterprise Linux 9 Benchmark v2.0.0"
                },
                {
                    "name": "CIS Rocky Linux 8 Benchmark v2.0.0"
                },
                {
                    "name": "CIS Rocky Linux 9 Benchmark v2.0.0"
                },
                {
                    "name": "CIS Ubuntu Linux 24.04 LTS Benchmark v1.0.0"
                }
            ]
        }
    }
}
```

Получение списка объектов сущности

Позволяет получить список объектов сущности определенного домена через запрос `Objects`.

URL: `{{url}}/api/evo/cms/bff/api/graphql`

Таблица 72. Параметры запроса `Objects`

Параметр	Тип данных	Обязателен	Описание
`domainId`	Строка	Да	ID домена
`entityId`	Строка	Да	ID сущности домена
`filter →`	Объект	Нет	Параметры фильтра
`→ ids`	Массив [Строка]	Нет	Массив UUID объектов
`→ query`	Строка	Нет	Сквозной поиск

`→ conditions`	Строка	Нет	Фильтр по значению полей
			
			Параметр является сериализованным JSON-объектом и имеет следующий формат:
			
			`"{\"${{логический оператор}}\":[{\"{{поле сущности}}\":{\"${{оператор сравнения}}\":\"{{значение поля}}\"}}]}"`
			
			Логические операторы: `or`, `and`
			
			Операторы сравнения: `eq` (=), `ne` (!=), `like`, `in`
			
			Пример 1313. Пример
			
			`"{\"$or\":[{\"name\":{\"$eq\":\"Хост\"}}]}"`

`pagination →`	Объект	Нет	Параметры пагинации

Параметр	Тип данных	Обязателен	Описание
→ limit	Целое число	Нет	Ограничение на количество возвращаемых результатов
→ offset	Целое число	Нет	Смещение
sorting →	Объект	Нет	Параметры сортировки
→ order	Строка	Нет	Порядок
→ orderBy	Строка	Нет	Поле сортировки

В ответ на запрос возвращаются следующие данные:

- objects: Параметры объекта.
- createdAt: Дата и время создания.
- creator: Кем создан объект.

  Пример 1314. Структура данных

  ```
  creator {
         displayName
         id

  }
  ```

- domainId: ID домена.
- entityId: ID сущности.
- id: UUID объекта.
- label: Имя объекта.
- tenantId: UUID тенанта.
- updatedAt: Дата и время изменения.
- updater: Кем изменен объект.

  Пример 1315. Структура данных

  ```
  updater {
         displayName
         id

  }
  ```

- values: Значения полей.
- pageInfo: Информация о странице (число объектов, пагинация).

Пример 1316. Структура данных

```
pageInfo {
     totalCount
     pagination {
           limit
           offset
     }

}
```

Пример 1317. Пример запроса `Objects`

```
query Objects($domainId: String!, $entityId: String!, $filter: ObjectsFilter,
$pagination: PaginationInput, $sorting: ObjectsSorting) {
    objects(domainId: $domainId, entityId: $entityId, filter: $filter,pagination:
$pagination, sorting: $sorting) {
            objects {
            domainId
            entityId
            label
            tenantId
            createdAt
            creator {
                id
                displayName
            }
            id
            updatedAt
            updater {
                displayName
                id
            }
            values
        }
    }
}
```

Пример 1318. Пример использования переменных

```
{
    "domainId": "isassets.rvision.ru",
    "entityId": "Device"
}
```

Пример 1319. Пример ответа

```
{
    "data": {
```

```
        "objects": {
            "objects": [
                {
                    "domainId": "isassets.rvision.ru",
                    "entityId": "Device",
                    "label": "alma85",
                    "tenantId": "00000000-0000-0000-0000-000000000000",
                    "createdAt": "2024-12-23T13:55:12.234Z",
                    "creator": {
                        "id": "00000000-0000-0000-0000-000000000000",
                        "displayName": "00000000-0000-0000-0000-000000000000"
                    },
                    "id": "c24c3851-edd0-485e-8c23-d1218b68ee7e",
                    "updatedAt": "2024-12-26T09:21:52.275Z",
                    "updater": {
                        "displayName": "admin",
                        "id": "4a84e42f-59d3-4736-9e55-41fc0e730dd2"
                    },
                    "values":
"{\"deviceGuid\":null,\"deviceName\":\"alma85\",\"criticalityDevice\":null,\"networkIn
terfaces\":null,\"ipAddress\":\"10.99.103.155\",\"mask\":null,\"macAdress\":null,\"dev
iceDomain\":\"rvlab.local\",\"FQDN\":null,\"operatingSystem\":null,\"virtualMachine\":
null,\"Hypervisor\":null,\"nodeType\":null,\"deviceStatus\":null,\"deviceGroupAssets\"
:null,\"deviceOwner\":null,\"assetTags\":null,\"deviceDescription\":\"щлжло\",\"port\"
:null,\"deviceSource\":null,\"inventoryDevice\":null,\"deviceScanDate\":null,\"deviceL
ocation\":null,\"deviceTz\":null}"
                }
            ]
        }
    }
}
```


Получение объекта сущности

Позволяет получить объект сущности определенного домена через запрос `object`.

URL: `/api/evo/cms/bff/api/graphql`

Таблица 73. Параметры запроса `object`

Параметр	Тип данных	Обязателен	Описание
`domainId`	Строка	Да	ID домена
`entityId`	Строка	Да	ID сущности домена
`id`	Строка	Да	UUID объекта

В ответ на запрос возвращаются следующие данные:

- `objects`: Параметры объекта.
- `createdAt`: Дата и время создания.

- `creator`: Кем создан объект.

  Пример 1320. Структура данных

  ```
  creator {
         displayName
         id

  }
  ```

- `domainId`: ID домена.
- `entityId`: ID сущности.
- `id`: UUID объекта.
- `label`: Имя объекта.
- `tenantId`: UUID тенанта.
- `updatedAt`: Дата и время изменения.
- `updater`: Кем изменен объект.

  Пример 1321. Структура данных

  ```
  updater {
         displayName
         id

  }
  ```

- `values`: Значения полей.
- `version`: Версия объекта.

Пример 1322. Пример запроса `object`

```
query object ($domainId: String!, $entityId: String!, $id: String!, $refFields:
[String!]) {
    object (domainId: $domainId, entityId: $entityId, id: $id, refFields: $refFields)
{
        createdAt
        domainId
        entityId
        id
        label
        tenantId
        updatedAt
        values
        version
        creator {
            displayName
            id
```

```
        }
        updater {
            displayName
            id
        }
    }
}
```

Пример 1323. Пример использования переменных

```
{
  "domainId": "isassets.rvision.ru",
  "entityId": "Device",
  "id": "c24c3851-edd0-485e-8c23-d1218b68ee7e"

}
```

Пример 1324. Пример ответа

```
{
    "data": {
        "object": {
            "createdAt": "2024-12-23T13:55:12.234Z",
            "domainId": "isassets.rvision.ru",
            "entityId": "Device",
            "id": "c24c3851-edd0-485e-8c23-d1218b68ee7e",
            "label": "alma85",
            "tenantId": "00000000-0000-0000-0000-000000000000",
            "updatedAt": "2024-12-26T09:21:52.275Z",
            "values":
"{\"deviceGuid\":null,\"deviceName\":\"alma85\",\"criticalityDevice\":null,\"networkIn
terfaces\":null,\"ipAddress\":\"10.99.103.155\",\"mask\":null,\"macAdress\":null,\"dev
iceDomain\":\"rvlab.local\",\"FQDN\":null,\"operatingSystem\":null,\"virtualMachine\":
null,\"Hypervisor\":null,\"nodeType\":null,\"deviceStatus\":null,\"deviceGroupAssets\"
:null,\"deviceOwner\":null,\"assetTags\":null,\"deviceDescription\":\"щлжло\",\"port\"
:null,\"deviceSource\":null,\"inventoryDevice\":null,\"deviceScanDate\":null,\"deviceL
ocation\":null,\"deviceTz\":null}",
            "version": 1,
            "creator": {
                "displayName": "00000000-0000-0000-0000-000000000000",
                "id": "00000000-0000-0000-0000-000000000000"
            },
            "updater": {
                "displayName": "admin",
                "id": "4a84e42f-59d3-4736-9e55-41fc0e730dd2"
            }
        }
    }
}
```

Получение связей объекта с другими объектами

Позволяет получить связи определенного объекта сущности с объектами сущностей текущего или другого домена через запрос `ObjectGraph`.

URL: `{{url}}/api/evo/cms/bff/api/graphql`

Таблица 74. Параметры запроса `ObjectGraph`

Параметр	Тип данных	Обязателен	Описание
`domainId`	Строка	Да	ID домена сущности
`entityId`	Строка	Да	ID сущности
`id`	Строка	Да	ID объекта
`level`	Целое число	Да	Максимальный уровень вложенности связи
`relations →`	Массив [Объект]	Да	Массив связей
`→ domainId`	Строка	Да	ID домена связанной сущности
`→ linkageId`	Строка	Да	ID типа связи. Пример: `"linkageId": "device_to_softwa re"`

В ответ на запрос возвращаются следующие данные:

- `graphConnections`: Параметры связи.

  Пример 1325. Структура данных

  ```
  graphConnections {
              domainId
              from
              id
              linkageId
              to
          }
  ```

Пример 1326. Пример запроса `ObjectGraph`

```
query ObjectGraph ($domainId: String!, $entityId: String!, $id: String!, $level: Int!,
$relations:[CmsObjectGraphRelation!]!) {
    objectGraph(
        domainId: $domainId, entityId: $entityId, id: $id,
        level: $level, relations: $relations) {
```

```
        graphConnections {
            domainId
            from
            id
            linkageId
            to
        }
    }
}
```

Пример 1327. Пример использования переменных

```
{
    "domainId": "isassets.rvision.ru",
    "entityId": "Device",
    "id": "c24c3851-edd0-485e-8c23-d1218b68ee7e",
    "level": 1,
    "relations": {
        "domainId": "isassets.rvision.ru",
        "linkageId": "device_to_software"
    }
}
```

Пример 1328. Пример ответа

```
{
    "data": {
        "objectGraph": {
            "graphConnections": [
                {
                    "domainId": "isassets.rvision.ru",
                    "from": "c24c3851-edd0-485e-8c23-d1218b68ee7e",
                    "id": "34da8d34-51ad-4659-bf7a-1b3a3e47308a",
                    "linkageId": "device_to_software",
                    "to": "84ed4fb5-9c32-46ff-aac1-ff9f4cba768a"
                },
                {
                    "domainId": "isassets.rvision.ru",
                    "from": "c24c3851-edd0-485e-8c23-d1218b68ee7e",
                    "id": "3849888b-44fb-43eb-859e-e936b6093b88",
                    "linkageId": "device_to_software",
                    "to": "38c8add3-5055-4cb2-9147-b1b27f729666"
                },
...
```

Создание объекта сущности

Позволяет создать объект определенной сущности через мутацию `createObject`.

URL: `{{url}}/api/evo/cms/bff/api/graphql`

Ключ	Описание
x-tenant-id	UUID тенанта, в котором находится сущность

Таблица 75. Параметры мутации `createObject`

Параметр	Тип данных	Обязателен	Описание
`input →`	Объект	Да	Параметры объекта
`→ domainId`	Строка	Да	ID домена сущности
`→ entityId`	Строка	Да	ID сущности

Параметр	Тип данных	Обязателен	Описание
→ `values`	Строка	Да	Значения полей, определенных схемой домена в формате сериализованного JSON-объекта. Пример: `"values": "{\"deviceGuid\": null,\"deviceName\":\"alma85\",\"criticalityDevice\":null,\"networkInterfaces\":null,\"ipAddress\":\"10.99.103.155\",\"mask\":null,\"macAdress\":null,\"deviceDomain\":\"rvlab.local\",\"FQDN\":null,\"operatingSystem\":null,\"virtualMachine\":null,\"Hypervisor\":null,\"nodeType\":null,\"deviceStatus\":null,\"deviceGroupAssets\":null,\"deviceOwner\":null,\"assetTags\":null,\"deviceDescription\":\"щлжло\",\"port\":null,\"deviceSource\":null,\"inventoryDevice\":null,\"deviceScanDate\":null,\"deviceLocation\":null,\"deviceTz\":null}"`

В ответ на мутацию возвращаются следующие данные:

- `createdAt`: Дата и время создания.
- `creator`: Кем создан объект.

Пример 1329. Структура данных

```
creator {
       displayName
       id

}
```

- `domainId`: ID домена.
- `entityId`: ID сущности.
- `id`: UUID объекта.
- `label`: Имя объекта.
- `tenantId`: UUID тенанта.
- `updatedAt`: Дата и время изменения.
- `values`: Значения полей.
- `version`: Версия объекта.

Пример 1330. Пример мутации `createObject`

```
mutation createObject ($input: CreateObjectInput!) {
    createObject (input: $input) {
        createdAt
        domainId
        entityId
        id
        label
        tenantId
        updatedAt
        values
        version
    }
}
```

Пример 1331. Пример использования переменных

```
{
  "input": {
    "domainId": "isassets.rvision.ru",
    "entityId": "Device",
    "values":
"{\"deviceGuid\":null,\"deviceName\":\"test\",\"criticalityDevice\":null,\"networkInte
rfaces\":null,\"ipAddress\":\"10.99.103.155\",\"mask\":null,\"macAdress\":null,\"devic
eDomain\":\"rvlab.local\",\"FQDN\":null,\"operatingSystem\":null,\"virtualMachine\":nu
ll,\"Hypervisor\":null,\"nodeType\":null,\"deviceStatus\":null,\"deviceGroupAssets\":n
ull,\"deviceOwner\":null,\"assetTags\":null,\"deviceDescription\":\"щлжло\",\"port\":n
ull,\"deviceSource\":null,\"inventoryDevice\":null,\"deviceScanDate\":null,\"deviceLoc
ation\":null,\"deviceTz\":null}"
```

```
  }
}
```

Пример 1332. Пример ответа

```
{
    "data": {
        "createObject": {
            "createdAt": "2025-01-14T12:43:40.379Z",
            "domainId": "isassets.rvision.ru",
            "entityId": "Device",
            "id": "3f571cae-267a-435b-9a0b-4b20b7505b23",
            "label": "test",
            "tenantId": "00000000-0000-0000-0000-000000000000",
            "updatedAt": "2025-01-14T12:43:40.379Z",
            "version": 1,
            "values":
"{\"deviceGuid\":null,\"deviceName\":\"test\",\"criticalityDevice\":null,\"networkInte
rfaces\":null,\"ipAddress\":\"10.99.103.155\",\"mask\":null,\"macAdress\":null,\"devic
eDomain\":\"rvlab.local\",\"FQDN\":null,\"operatingSystem\":null,\"virtualMachine\":nu
ll,\"Hypervisor\":null,\"nodeType\":null,\"deviceStatus\":null,\"deviceGroupAssets\":n
ull,\"deviceOwner\":null,\"assetTags\":null,\"deviceDescription\":\"щлжло\",\"port\":n
ull,\"deviceSource\":null,\"inventoryDevice\":null,\"deviceScanDate\":null,\"deviceLoc
ation\":null,\"deviceTz\":null}"

        }
    }
}
```

Удаление объекта сущности

Позволяет удалить объект определенной сущности через мутацию `deleteObject`.

URL: `{{url}}/api/evo/cms/bff/api/graphql`

Ключ	Описание
x-tenant-id	UUID тенанта, в котором находится сущность

Таблица 76. Параметры мутации `deleteObject`

Параметр	Тип данных	Обязателен	Описание
`input →`	Объект	Да	Параметры объекта
`→ domainId`	Строка	Да	ID домена сущности
`→ entityId`	Строка	Да	ID сущности
`→ id`	Строка	Да	UUID объекта

В ответ на мутацию возвращаются следующие данные:

- `id`: UUID объекта.

Пример 1333. Пример мутации `deleteObject`

```
mutation deleteObject ($input: DeleteObjectInput!) {
    deleteObject (input: $input) {
        id
    }
}
```

Пример 1334. Пример использования переменных

```
{
  "input": {
    "domainId": "isassets.rvision.ru",
    "entityId": "Device",
    "id": "c24c3851-edd0-485e-8c23-d1218b68ee7e"
  }
}
```

Редактирование объекта сущности

Позволяет редактировать объект определенной сущности через мутацию `updateObject`.

URL: `{{url}}/api/evo/cms/bff/api/graphql`

Ключ	Описание
x-tenant-id	UUID тенанта, в котором находится сущность

Таблица 77. Параметры мутации `updateObject`

Параметр	Тип данных	Обязателен	Описание
`input →`	Объект	Да	Параметры объекта
`→ domainId`	Строка	Да	ID домена сущности
`→ entityId`	Строка	Да	ID сущности

Параметр	Тип данных	Обязателен	Описание
→ `values`	Строка	Да	Значения полей, определенных схемой домена в формате сериализованного JSON-объекта. Пример: `"values": "{\"deviceGuid\": null,\"deviceName\":\"alma85\",\"criticalityDevice\":null,\"networkInterfaces\":null,\"ipAddress\":\"10.99.103.155\",\"mask\":null,\"macAdress\":null,\"deviceDomain\":\"rvlab.local\",\"FQDN\":null,\"operatingSystem\":null,\"virtualMachine\":null,\"Hypervisor\":null,\"nodeType\":null,\"deviceStatus\":null,\"deviceGroupAssets\":null,\"deviceOwner\":null,\"assetTags\":null,\"deviceDescription\":\"щлжло\",\"port\":null,\"deviceSource\":null,\"inventoryDevice\":null,\"deviceScanDate\":null,\"deviceLocation\":null,\"deviceTz\":null}"`

В ответ на мутацию возвращаются следующие данные:

- `createdAt`: Дата и время создания.
- `creator`: Кем создан объект.

Пример 1335. Структура данных

```
creator {
       displayName
       id

}
```

- `updater`: Кем изменен объект.

Пример 1336. Структура данных

```
updater {
       displayName
       id

}
```

- `domainId`: ID домена.
- `entityId`: ID сущности.
- `id`: UUID объекта.
- `label`: Имя объекта.
- `tenantId`: UUID тенанта.
- `updatedAt`: Дата и время изменения.
- `values`: Значения полей.
- `version`: Версия объекта.

Пример 1337. Пример мутации `updateObject`

```
mutation updateObject ($input: UpdateObjectInput!) {
    updateObject (input: $input) {
        createdAt
        domainId
        entityId
        id
        label
        tenantId
        updatedAt
        values
        version
    }
}
```

Пример 1338. Пример использования переменных

```
{
```

```
  "input": {
    "domainId": "isassets.rvision.ru",
    "entityId": "Device",
    "id": "3f571cae-267a-435b-9a0b-4b20b7505b23",
    "values":
"{\"deviceGuid\":null,\"deviceName\":\"test1\",\"criticalityDevice\":null,\"networkInt
erfaces\":null,\"ipAddress\":\"10.99.103.155\",\"mask\":null,\"macAdress\":null,\"devi
ceDomain\":\"rvlab.local\",\"FQDN\":null,\"operatingSystem\":null,\"virtualMachine\":n
ull,\"Hypervisor\":null,\"nodeType\":null,\"deviceStatus\":null,\"deviceGroupAssets\":
null,\"deviceOwner\":null,\"assetTags\":null,\"deviceDescription\":\"щлжло\",\"port\":
null,\"deviceSource\":null,\"inventoryDevice\":null,\"deviceScanDate\":null,\"deviceLo
cation\":null,\"deviceTz\":null}"
  }
}
```

Пример 1339. Пример ответа

```
{
    "data": {
        "updateObject": {
            "createdAt": "2025-01-14T12:43:40.379Z",
            "domainId": "isassets.rvision.ru",
            "entityId": "Device",
            "id": "3f571cae-267a-435b-9a0b-4b20b7505b23",
            "label": "test1",
            "tenantId": "00000000-0000-0000-0000-000000000000",
            "updatedAt": "2025-01-14T15:44:58.087Z",
            "values":
"{\"deviceGuid\":null,\"deviceName\":\"test1\",\"criticalityDevice\":null,\"networkInt
erfaces\":null,\"ipAddress\":\"10.99.103.155\",\"mask\":null,\"macAdress\":null,\"devi
ceDomain\":\"rvlab.local\",\"FQDN\":null,\"operatingSystem\":null,\"virtualMachine\":n
ull,\"Hypervisor\":null,\"nodeType\":null,\"deviceStatus\":null,\"deviceGroupAssets\":
null,\"deviceOwner\":null,\"assetTags\":null,\"deviceDescription\":\"щлжло\",\"port\":
null,\"deviceSource\":null,\"inventoryDevice\":null,\"deviceScanDate\":null,\"deviceLo
cation\":null,\"deviceTz\":null}",
            "version": 2,
            "creator": {
                "displayName": "admin",
                "id": "4a84e42f-59d3-4736-9e55-41fc0e730dd2"
            },
            "updater": {
                "displayName": "admin",
                "id": "4a84e42f-59d3-4736-9e55-41fc0e730dd2"
            }
        }
    }
}
```

Создание связи объекта

Позволяет создать связь объекта с объектами текущего или других доменов через мутацию `createObjectLinks`.

URL: `{{url}}/api/evo/cms/bff/api/graphql`

Ключ	Описание
x-tenant-id	UUID тенанта, в котором находится сущность

Таблица 78. Параметры мутации `createObjectLinks`

Параметр	Тип данных	Обязателен	Описание
`input →`	Объект	Да	Параметры объекта
`→ domainId`	Строка	Да	ID домена
`→ entityId`	Строка	Да	ID сущности
`→ objectId`	Строка	Да	ID объекта
`→ inputs →`	Массив [Объект]	Да	Массив связей
`→ → domainId`	Строка	Да	ID домена связанного объекта
`→ → linkageId`	Строка	Да	ID типа связи. Пример: `"linkageId": "device_to_software"`
`objectId`	Строка	Да	ID связанного объекта

В ответ на мутацию возвращаются следующие данные:

- `createdAt`: Дата и время создания связи.
- `creator`: Кем создан объект.

  Пример 1340. Структура данных

  ```
  creator {
         displayName
         id

  }
  ```

- `domainId`: ID домена.
- `id`: UUID связи.
- `linkageId`: ID связи.

- `side1`: Параметры объекта 1.

  Пример 1341. Структура данных

  ```
  side1 {
              domainId
              entityId
              id
              tenantId
              version
          }
  ```

- `side2`: Параметры объекта 2.

  Пример 1342. Структура данных

  ```
  side2 {
              domainId
              entityId
              id
              tenantId
              version
          }
  ```

Пример 1343. Пример мутации `createObjectLinks`

```
mutation createObjectLinks ($input: CreateObjectLinksInput!) {
    createObjectLinks (input: $input) {
        createdAt
        domainId
        id
        linkageId
        side1 {
            domainId
            entityId
            id
            tenantId
            version
        }
        side2 {
            domainId
            entityId
            id
            tenantId
            version
        }
    }
}
```

Пример 1344. Пример использования переменных

```
{
  "input": {
    "domainId": "isassets.rvision.ru",
    "entityId": "Device",
    "objectId": "3f571cae-267a-435b-9a0b-4b20b7505b23",
    "inputs": {
      "domainId": "isassets.rvision.ru",
      "linkageId": "device_to_software",
      "objectId": "3e72182c-6fae-443f-bcaa-9309afece8c5"
    }
  }
}
```

Пример 1345. Пример ответа

```
{
    "data": {
        "createObjectLinks": [
            {
                "createdAt": "2025-01-14T15:47:33.326Z",
                "domainId": "isassets.rvision.ru",
                "id": "69ac7798-e1d3-409d-87a3-f7a88346060c",
                "linkageId": "device_to_software",
                "side1": {
                    "domainId": "isassets.rvision.ru",
                    "entityId": "Device",
                    "id": "3f571cae-267a-435b-9a0b-4b20b7505b23",
                    "tenantId": "00000000-0000-0000-0000-000000000000",
                    "version": 2
                },
                "side2": {
                    "domainId": "isassets.rvision.ru",
                    "entityId": "Software",
                    "id": "3e72182c-6fae-443f-bcaa-9309afece8c5",
                    "tenantId": "00000000-0000-0000-0000-000000000000",
                    "version": 3
                }
            }
        ]
    }
}
```

Удаление связи объекта

Позволяет удалить связь объекта с объектами текущего или других доменов через мутацию `deleteObjectLinks`.

URL: `{{url}}/api/evo/cms/bff/api/graphql`

Таблица 79. Параметры мутации `deleteObjectLinks`

Параметр	Тип данных	Обязателен	Описание
`input →`	Объект	Да	Параметры объекта
`→ domainId`	Строка	Да	ID домена
`→ entityId`	Строка	Да	ID сущности
`→ objectId`	Строка	Да	ID объекта
`→ inputs →`	Массив [Объект]	Да	Массив связей
`→ → domainId`	Строка	Да	ID домена связанного объекта
`→ → linkageId`	Строка	Да	ID типа связи. Пример: `"linkageId": "device_to_softwa re"`
`→ → objectIds`	Массив [Строка]	Да	Массив ID связанных объектов

В ответ на мутацию возвращаются следующие данные:

- `domainId`: ID домена.
- `ids`: UUID связей.
- `linkageId`: ID связи.

Пример 1346. Пример мутации `deleteObjectLinks`

```
mutation deleteObjectLinks ($input: DeleteObjectLinksInput!) {
    deleteObjectLinks (input: $input) {
        domainId
        ids
        linkageId
    }
}
```

Пример 1347. Пример использования переменных

```
{
  "input": {
    "domainId": "isassets.rvision.ru",
    "entityId": "Device",
    "objectId": "3f571cae-267a-435b-9a0b-4b20b7505b23",
    "inputs": {
      "domainId": "isassets.rvision.ru",
```

```
      "linkageId": "device_to_software",
      "objectIds": ["3e72182c-6fae-443f-bcaa-9309afece8c5"]
    }
  }
}
```

Пример 1348. Пример ответа

```
{
    "data": {
        "deleteObjectLinks": [
            {
                "domainId": "isassets.rvision.ru",
                "ids": [
                    "69ac7798-e1d3-409d-87a3-f7a88346060c"
                ],
                "linkageId": "device_to_software"
            }
        ]
    }
}
```

Получение учетной записи

Позволяет получить учетную запись через запрос `getAccount`.

URL: `{{url}}/api/evo/vm/bff/graphql`

Таблица 80. Параметры запроса `getAccount`

Параметр	Тип данных	Обязателен	Описание
`id`	Строка	Да	UUID учетной записи
`tenantId`	Строка	Да	UUID тенанта

В ответ на запрос возвращаются следующие данные:

- `accountType`: Тип учетной записи.
- `active`: Учетная запись включена.
- `description`: Описание.
- `login`: Логин.
- `password`: Пароль.
- `name`: Название учетной записи.
- `dbAdditionalSettings`: Параметры учетной записи СУБД.

  Пример 1349. Структура данных

  ```
  dbAdditionalSettings {
  ```

```
        database
        instance
        port
        serviceName
        type
    }
```

- `privileges`: Повышение привилегий:
  - Sudo (для Linux).
  - Enable (для Cisco).
  - Expert (для Checkpoint).
  - Super (для Linux).

Пример 1350. Структура данных

```
privileges {
    enable {
        enabled: Boolean
        password: String
        level: Int
    }
    expert {
        enabled: Boolean
        password: String
    }
    sudo {
        enabled: Boolean
        password: String
    }
    super {
        enabled: Boolean
        password: String
        level: Int
    }
}
```

- `sshAuthenticationMethod`: Метод аутентификации:
  - SSH_AUTH_METHOD_PASSWORD — Пароль.
  - SSH_AUTH_METHOD_PRIVATE_KEY — Публичный ключ.
- `sshPrivateKeyFilename`: Название файла закрытого ключа.
- `sshPrivateKey`: Закрытый ключ.
- `sshPrivateKeyPassphrase`: Пароль закрытого ключа.
- `tenantId`: UUID тенанта.

Пример 1351. Пример запроса `getAccount`

```
query getAccount ($id: String!, $organizationId: String) {
    getAccount (id: $id, organizationId: $organizationId) {
        accountType
        active
        description
        id
        isPasswordSet
        login
        name
        sshAuthenticationMethod
        sshPrivateKeyFilename
        sshPublicKey
        privileges {
            enable {
                enabled
                isPasswordSet
                level
            }
            expert {
                enabled
                isPasswordSet
            }
            sudo {
                enabled
                isPasswordSet
            }
            super {
                enabled
                isPasswordSet
                level
            }
        }
        tenantId
    }
}
```

Пример 1352. Пример использования переменных

```
{
  "id": "007bf0d4-47a3-48c6-a884-fa52b578ef78"
}
```

Пример 1353. Пример ответа

```
{
    "data": {
        "getAccount": {
            "accountType": "TYPE_OF_ACCOUNT_SSH",
```

```
            "active": true,
            "description": "autotest",
            "dbAdditionalSettings": null,
            "id": "00325cae-0722-47bc-aff5-33a3960d89cb",
            "isPasswordSet": true,
            "login": "root",
            "name": "autotest",
            "sshAuthenticationMethod": "SSH_AUTH_METHOD_PASSWORD",
            "sshPrivateKeyFilename": "",
            "sshPublicKey": "",
            "tenantId": "f88b30ff-7862-489a-a6dd-c82c23e5ea21",
            "privileges": {
                "enable": {
                    "enabled": false,
                    "isPasswordSet": false,
                    "level": 0
                },
                "expert": {
                    "enabled": false,
                    "isPasswordSet": false
                },
                "sudo": {
                    "enabled": false,
                    "isPasswordSet": false
                },
                "super": {
                    "enabled": false,
                    "isPasswordSet": false,
                    "level": 0
                }
            }
        }
    }
}
```


Проверка учетной записи на устройстве

Позволяет осуществлять проверку учетной записи на устройстве через запрос `checkAccount`.

URL: `{{url}}/api/evo/vm/bff/graphql`

Таблица 81. Параметры запроса `checkAccount`

Параметр	Тип данных	Обязателен	Описание
`account →`	Объект	Да	Параметры учетной записи для проверки
`→ id`	Строка	Нет	UUID учетной записи
`→ login`	Строка	Да	Логин учетной записи
`→ password`	Строка	Нет	Пароль учетной записи

Параметр	Тип данных	Обязателен	Описание
→ sshAuthenticationMethod	Строка	Нет	Способ аутентификации
→ sshPrivateKey	Строка	Нет	Пароль закрытого ключа
→ sshPublicKey	Строка	Нет	—
→ type	Строка	Да	Тип учетной записи
host →	Объект	Да	—
→ address	Строка	Да	Имя или IP устройства
→ port	Целое число	Да	Порт подключения
spaceId	Строка	Да	UUID пространства
workloadId	Строка	Да	UUID сканера

В ответ на запрос возвращаются следующие данные:

- error: Текст ошибки запроса.
- success: Статус запроса.

Пример 1354. Пример запроса checkAccount

```
query checkAccount ($account: CheckAccountParams!, $host: CheckAccountHostParams!,
$spaceId: String!, $workloadId: String!) {
    checkAccount (account: $account, host: $host, spaceId: $spaceId, workloadId:
$workloadId) {
        error
        success
    }
}
```

Пример 1355. Пример использования переменных

```
{
  "account": {
    "login": "login@rvlab.ru",
    "password": "password",
    "type": "TYPE_OF_ACCOUNT_WINDOWS"
  },
  "host": {
    "address": "admin.rvlab.local",
    "port": 22
  },
  "spaceId": "space-2",
  "workloadId": "vm-scanners-1"
}
```

Получение списка учетных записей

Позволяет получить список учетных записей через запрос `findAccounts`.

URL: `{{url}}/api/evo/vm/bff/graphql`

Таблица 82. Параметры запроса `findAccounts`

Параметр	Тип данных	Обязателен	Описание
`filter →`	Объект	Нет	Параметры фильтра
`→ name`	Строка	Нет	Название задачи
`→ query`	Строка	Нет	Сквозной поиск по полям: • `login`; • `name`.
`→ active`	булево	Нет	Учетная запись включена
`→ ids`	Массив [Строка]	Нет	Массив UUID учетных записей
`→ tenantId`	Строка	Нет	UUID тенанта
`pagination →`	Объект	Нет	Параметры пагинации
`→ limit`	Целое число	Нет	Ограничение на количество возвращаемых результатов
`→ offset`	Целое число	Нет	Смещение
`sorting →`	Объект	Нет	Параметры сортировки
`→ order`	Строка	Нет	Порядок
`→ orderBy`	Строка	Нет	Поле сортировки: • ORDER_BY_ACCOUNT_TYPE — Тип УЗ • ORDER_BY_ACTIVE — Статус • ORDER_BY_LOGIN — Логин • ORDER_BY_NAME — Название

В ответ на запрос возвращаются следующие данные:

- `accounts`: Текст ошибки запроса.

Пример 1356. Структура данных

```
accounts {
    accountType
    active
    description
    dbAdditionalSettings {
        database
        instance
        port
        serviceName
        type
    }
    id
    isPasswordSet
    login
    name
    organizationId
    sshAuthenticationMethod
    sshPrivateKeyFilename
    sshPublicKey
    tenantId
    privileges {
        enable {
            enabled
            isPasswordSet
            level
        }
        expert {
            enabled
            isPasswordSet
        }
        sudo {
            enabled
            isPasswordSet
        }
        super {
            enabled
            isPasswordSet
            level
        }
    }
}
```

- `pageInfo`: Статус запроса.

Пример 1357. Структура данных

```
pageInfo {
    totalCount
    pagination {
```

```
        limit
        offset
    }
}
```

Пример 1358. Пример запроса `findAccounts`

```
query findAccounts  ($filter: AccountsFilter, $pagination: PaginationInput, $sorting:
AccountsSorting) {
    findAccounts  (filter: $filter, pagination: $pagination, sorting: $sorting) {
        accounts {
            accountType
            active
            description
            dbAdditionalSettings {
                database
                instance
                port
                serviceName
                type
            }
            id
            isPasswordSet
            login
            name
            sshAuthenticationMethod
            sshPrivateKeyFilename
            sshPublicKey
            tenantId
            privileges {
                enable {
                    enabled
                    isPasswordSet
                    level
                }
                expert {
                    enabled
                    isPasswordSet
                }
                sudo {
                    enabled
                    isPasswordSet
                }
                super {
                    enabled
                    isPasswordSet
                    level
                }
            }
        }
```

```
    }
}
```

Пример 1359. Пример использования переменных

```
{
    "filter":
    {
        "name": "MSSQL"
    }
}
```

Пример 1360. Пример ответа

```
{
    "data": {
        "findAccounts": {
            "accounts": [
                {
                    "accountType": "TYPE_OF_ACCOUNT_DATABASE",
                    "active": true,
                    "description": "",
                    "dbAdditionalSettings": {
                        "database": "master",
                        "instance": "MSSQLSERVER",
                        "port": 1433,
                        "serviceName": null,
                        "type": "TYPE_OF_DATABASE_MSSQL"
                    },
                    "id": "14f6e347-8b7c-4f69-8098-4903f93b693a",
                    "isPasswordSet": true,
                    "login": "admin",
                    "name": "MSSQL",
                    "sshAuthenticationMethod": "SSH_AUTH_METHOD_PASSWORD",
                    "sshPrivateKeyFilename": "",
                    "sshPublicKey": "",
                    "tenantId": "f88b30ff-7862-489a-a6dd-c82c23e5ea21",
                    "privileges": {
                        "enable": {
                            "enabled": false,
                            "isPasswordSet": false,
                            "level": 0
                        },
                        "expert": {
                            "enabled": false,
                            "isPasswordSet": false
                        },
                        "sudo": {
                            "enabled": false,
                            "isPasswordSet": false
```

```
                        },
                        "super": {
                            "enabled": false,
                            "isPasswordSet": false,
                            "level": 0
                        }
                    }
                }
            ]
        }
    }
}
```

Создание учетной записи

Позволяет создать учетную запись через мутацию `createAccount`.

URL: `{{url}}/api/evo/vm/bff/graphql`

Таблица 83. Параметры мутации `createAccount`

Параметр	Тип данных	Обязателен	Описание
`input →`	Объект	Да	Параметры учетной записи
`→ accountType`	Строка	Да	Тип учетной записи
`→ active`	Булево	Да	Учетная запись включена
`→ description`	Строка	Да	Описание
`→ dbAdditionalSettings →`	Объект	Да, если Тип учетной записи = СУБД	Параметры учетной записи СУБД
`→ → database`	Строка	Да, если Тип учетной записи = СУБД	Название базы данных
`→ → instance`	Строка	Да, если Тип учетной записи = СУБД, Тип СУБД = MS SQL	Экземпляр
`→ → port`	Целое число	Да, если Тип учетной записи = СУБД	Порт
`→ → serviceName`	Строка	Да, если Тип учетной записи = СУБД, Тип СУБД = Oracle	Имя службы
`→ → type`	Строка	Да, если Тип учетной записи = СУБД	Тип СУБД
`→ login`	Строка	Да	Логин

Параметр	Тип данных	Обязателен	Описание
→ password	Строка	Да, если Метод аутентификации = Пароль	Пароль
→ name	Строка	Да	Название учетной записи
→ privileges →	Объект	Да	Повышение привилегий
→ → enable	Объект *Пример 1361. Структура данных* `privileges { enable { enabled: Boolean password: String level: Int } }`	Enable (для Cisco)	Нет
→ → expert	Объект *Пример 1362. Структура данных* `expert { enabled: Boolean password: String } }`	Expert (для Checkpoint)	Нет
→ → sudo	Объект *Пример 1363. Структура данных* `sudo { enabled: Boolean password: String } }`	Sudo (для Linux)	Нет

Параметр	Тип данных	Обязателен	Описание
→ → super	Объект *Пример 1364. Структура данных* `super { enabled: Boolean password: String level: Int }`	Super (для Linux)	Нет
→ sshAuthenticationMethod	Строка	Нет	Метод аутентификации: • SSH_AUTH_METHOD_PASSWORD – Пароль • SSH_AUTH_METHOD_PRIVATE_KEY – Публичный ключ
→ sshPrivateKeyFilename	Строка	Нет	Название файла закрытого ключа
→ sshPrivateKey	Строка	Нет	Закрытый ключ
→ sshPrivateKeyPassphrase	Строка	Нет	Пароль закрытого ключа
→ tenantId	Строка	Да	Строка UUID

В ответ на запрос возвращаются следующие данные:

- error: Текст ошибки запроса.
- success: Статус запроса.
- data: Параметры учетной записи.

Пример 1365. Структура данных

```
data {
    accountType
    active
    description
    dbAdditionalSettings {
        database
        instance
        port
        serviceName
```

```
        type
    }
    id
    isPasswordSet
    login
    name
    sshAuthenticationMethod
    sshPrivateKeyFilename
    sshPublicKey
    tenantId
    privileges {
        enable {
            enabled
            isPasswordSet
            level
        }
        expert {
            enabled
            isPasswordSet
        }
        sudo {
            enabled
            isPasswordSet
        }
        super {
            enabled
            isPasswordSet
            level
        }
    }
    tenantId
}
```

Пример 1366. Пример мутации `createAccount`

```
mutation createAccount ($input: CreateAccountInput!) {
    createAccount (input: $input) {
        error
        success
        tenantId
    }
}
```

Пример 1367. Пример использования переменных

```
{
  "input": {
    "accountType": "TYPE_OF_ACCOUNT_WINDOWS",
    "active": true,
    "description": "",
```

```
    "login": "admin",
    "name": "admin",
    "password": "admin",
    "privileges": {
      "enable": {
        "enabled": false,
        "level": 0,
        "password": ""
      },
      "expert": {
        "enabled": false,
        "password": ""
      },
      "sudo": {
        "enabled": false,
        "password": ""
      },
      "super": {
        "enabled": false,
        "level": 0,
        "password": ""
      }
    }
    "tenantId": "f88b30ff-7862-489a-a6dd-c82c23e5ea21",
  }
}
```

Удаление учетной записи

Позволяет удалить учетную запись через мутацию `deleteAccount`.

URL: `{{url}}/api/evo/vm/bff/graphql`

Таблица 84. Параметры мутации `deleteAccount`

Параметр	Тип данных	Обязателен	Описание
`delete →`	Объект	Да	—
`→ id`	Строка	Да	UUID учетной записи

В ответ на запрос возвращаются следующие данные:

- `error`: Текст ошибки запроса.
- `success`: Статус запроса.

Пример 1368. Пример мутации `deleteAccount`

```
mutation deleteAccount ($delete: DeleteAccountInput!) {
    deleteAccount (delete: $delete) {
        error
        success
```

```
    }
}
```

Пример 1369. Пример использования переменных

```
{
  "delete": {
    "id": "02210d31-1cf1-4501-abc1-94484f011348"
  }
}
```

Редактирование учетной записи

Позволяет отредактировать учетную запись через мутацию `updateAccount`.

URL: `{{url}}/api/evo/vm/bff/graphql`

Таблица 85. Параметры мутации `updateAccount`

Параметр	Тип данных	Обязателен	Описание
`update →`	Объект	Да	Параметры учетной записи
`→ active`	Булево	Нет	Учетная запись включена
`→ description`	Строка	Нет	Описание
`→ dbAdditionalSettings →`	Объект	Да, если Тип учетной записи = СУБД	Параметры учетной записи СУБД
`→ → database`	Строка	Да, если Тип учетной записи = СУБД	Название базы данных
`→ → instance`	Строка	Да, если Тип учетной записи = СУБД	Экземпляр (Тип СУБД = MS SQL)
`→ → port`	Целое число	Да, если Тип учетной записи = СУБД	Порт
`→ → serviceName`	Строка	Да, если Тип учетной записи = СУБД	Имя службы (Тип СУБД = Oracle)
`→ → type`	Строка	Да, если Тип учетной записи = СУБД	Тип СУБД
`→ id`	Строка	Да	UUID задачи
`→ login`	Строка	Нет	Логин
`→ password`	Строка	Нет	Пароль
`→ name`	Строка	Нет	Название

Параметр	Тип данных	Обязателен	Описание
→ privileges	Объект + *Пример 1370. Структура данных* `privileges { enable { enabled: Boolean password: String level: Int } expert { enabled: Boolean password: String } sudo { enabled: Boolean password: String } super { enabled: Boolean password: String level: Int } }`	Да	Повышение привилегий: • Sudo (для Linux) • Enable (для Cisco) • Expert (для Checkpoint) • Super (для Linux)
→ sshAuthenticationMethod	Строка	Нет	Метод аутентификации: • SSH_AUTH_METHOD_PASSWORD – Пароль • SSH_AUTH_METHOD_PRIVATE_KEY – Публичный ключ
→ sshPrivateKeyFilename	Строка	Нет	Название файла закрытого ключа

Параметр	Тип данных	Обязателен	Описание
→ sshPrivateKey	Строка	Нет	Закрытый ключ
→ sshPrivateKeyPassphrase	Строка	Нет	Пароль закрытого ключа

В ответ на запрос возвращаются следующие данные:

- error: Текст ошибки запроса.
- success: Статус запроса.
- data: Параметры учетной записи

Пример 1371. Структура данных

```
data {
    accountType
    active
    description
    dbAdditionalSettings {
        database
        instance
        port
        serviceName
        type
    }
    id
    isPasswordSet
    login
    name
    sshAuthenticationMethod
    sshPrivateKeyFilename
    sshPublicKey
    organizationId
    privileges {
        enable {
            enabled
            isPasswordSet
            level
        }
        expert {
            enabled
            isPasswordSet
        }
        sudo {
            enabled
            isPasswordSet
        }
        super {
            enabled
            isPasswordSet
```

```
            level
        }
    }
}
```

Пример 1372. Пример мутации `updateAccount`

```
mutation updateAccount ($input: UpdateAccountInput!) {
    updateAccount (update: $update) {
        error
        success
    }
}
```

Пример 1373. Пример использования переменных

```
{
  "input": {
    "active": true,
    "description": "",
    "dbAdditionalSettings": null,
    "login": "admin",
    "name": "admin",
    "password": "admin",
    "privileges": {
      "enable": {
        "enabled": false,
        "level": 0,
        "password": ""
      },
      "expert": {
        "enabled": false,
        "password": ""
      },
      "sudo": {
        "enabled": false,
        "password": ""
      },
      "super": {
        "enabled": false,
        "level": 0,
        "password": ""
      }
    }
  }
}
```

Получение списка групп агентов

Позволяет получить список групп агентов через запрос `GetAgentScanGroup`.

В ответ на запрос возвращаются следующие данные:

- `id`: UUID группы агентов .
- `name`: Имя группы агентов.

Пример 1374. Пример запроса `GetAgentScanGroup`

```
query GetAgentScanGroup {
    getAgentScanGroup {
        data {
            id
            name
        }
    }
}
```

Пример 1375. Пример ответа

```
    "data": {
        "getAgentScanGroup": {
            "data": [
                {
                    "id": "e9bac87a-9ab9-4e93-bf4d-f27f1cffba03",
                    "name": "000real-agents"
                },
                {
                    "id": "44a756d1-a97d-4a10-96a7-233fed798cb6",
                    "name": "added-new-group"
                },
...
```

Получение списка агентов

Позволяет получить список агентов через запрос `agents`.

URL: `{{url}}/api/evo/ep/bff/graphql`

Таблица 86. Параметры запроса `agents`

Параметр	Тип данных	Обязателен	Описание
`filter →`	Объект	Нет	Параметры пагинации
`→ id`	Массив [Строка]	Нет	Массив UUID агентов

Параметр	Тип данных	Обязателен	Описание
→ status	Массив [Строка]	Нет	Статус агента: • REGISTERED — Зарегистрирован. • ONLINE — Онлайн. • DISABLED — Выключен. • OFFLINE — Оффлайн.
→ agentGroupId	Массив [Строка]	Нет	Массив UUID групп агентов, в которых состоят агенты
→ osType	Массив [Строка]	Нет	Тип ОС
→ search	Строка	Нет	Строка поиска
→ version	Строка	Нет	Версия агента Пример: `"version": "rpoint v1.10.4.5"`
→ registeredAtGte	Строка	Нет	Дата регистрации: от (в формате Datetime)
→ registeredAtLte	Строка	Нет	Дата регистрации: до (в формате Datetime)
→ lastActivity	Строка	Нет	Дата последней активности (в формате Datetime)
options →	Объект	Нет	—
→ limit	Целое число	Нет	Ограничение на количество возвращаемых результатов
→ offset	Целое число	Нет	Смещение
→ sort →	Объект	Нет	Параметры сортировки
→ → by	Строка	Нет	Поле сортировки
→ → order	Строка	Нет	Порядок

В ответ на запрос возвращаются следующие данные:

- id: UUID агента.

- `name`: Название агента.
- `machineId`: UUID машины.
- `hostname`: Имя хоста.
- `groupConfigStatus`: Статус группы агентов.
- `ip`: IP адреса хоста.
- `registeredAt`: Дата регистрации.
- `lastAliveAt`: Дата последней активности.
- `status`: Статус агента.
- `version`: Версия агента.
- `os`: Параметры ОС.

Пример 1376. Структура данных

```
os {
    major
    minor
    name
    platform
    version
    osType
    arch
    build
}
```

- `groups`: Параметры группы агентов, в которой состоит агент.

Пример 1377. Структура данных

```
groups {
    id
    name
    autoAssignment
    createdAt
    updatedAt
    agentCount
    vrlExpressions
}
```

- `inventory`: Данные инвентаризации.

Пример 1378. Структура данных

```
inventory {
    users
    containers
    processes
```

```
    openPorts
}
```

Пример 1379. Пример запроса `agents`

```
query agents ($filter: AgentFindFilter, $options: AgentFindOptions) {
    agents (filter: $filter, options: $options) {
        id
        name
        machineId
        hostname
        groupConfigStatus
        ip
        registeredAt
        lastAliveAt
        status
        version
        os {
            major
            minor
            name
            platform
            version
            osType
            arch
            build
        }
        groups {
            id
            name
        }
    }
}
```

Пример 1380. Пример использования переменных

```
{
  "filter": {},
  "options": {
    "limit": 20
  }
}
```

Пример 1381. Пример ответа

```
{
    "data": {
        "agents": [
            {
```

```
                "id": "010afe48-9f88-4256-afc4-2a28cea5f2eb",
                "name": "accomplished-niece.info",
                "machineId": "a4b39235-e4c9-4dfa-9062-ec6aa9603d04",
                "hostname": "accomplished-niece.info",
                "groupConfigStatus": "SYNCED",
                "ip": [
                    "71.199.20.179",
                    "53.1.163.162"
                ],
                "registeredAt": "2022-06-12T20:41:32.458Z",
                "lastAliveAt": "2025-01-22T21:25:47.701Z",
                "status": "OFFLINE",
                "version": "rpoint v1.10.4.5",
                "os": {
                    "major": "8",
                    "minor": "2",
                    "name": "Linux",
                    "platform": "rhel",
                    "version": "8.2",
                    "osType": "LINUX",
                    "arch": "x86_64",
                    "build": ""
                },
                "groups": [
                    {
                        "id": "ebf7445e-5fbc-4b25-85e6-2a60c0c1dde4",
                        "name": "one-hundred"
                    }
                ]
            },
 ...
```


Обработка ошибок API

В процессе взаимодействия с API системы могут возникать различные ошибки. Ниже приведен список возможных ошибок с описаниями и рекомендациями по их устранению.

Код ошибки	Описание
AUTH.ISSUE_TOKEN.TOKEN_ALREADY_EXIST	Попытка создать токен для пользователя, для которого ранее уже был создан токен. Убедитесь, что предыдущий токен не активен или используйте его для работы с API.
COLLECTOR.DELETE.R_SPACE_ERROR	Произошла ошибка при удалении коллектора. Обратитесь в службу поддержки для устранения проблемы.
COLLECTOR.DISABLE.ALREADY_DISABLED	Попытка отключения уже отключенного коллектора. Проверьте текущее состояние коллектора перед его отключением.

Код ошибки	Описание
COLLECTOR.ENABLE.ALREADY_ENABLED	Попытка включения уже включенного коллектора. Убедитесь, что коллектор отключен.
COLLECTOR.IMPORT.INCORRECT_DATA	Данные импортируемых коллекторов некорректны. Проверьте корректность данных коллекторов перед их импортом.
COLLECTOR_ENRICHMENT_TABLE.CREATE.ALREADY_CREATED	Попытка добавления таблицы обогащения в коллектор, в который уже была добавлена эта таблица обогащения. Убедитесь, что таблица обогащения еще не добавлена в коллектор
DASHBOARD.IMPORT.INCORRECT_DATA	Данные импортируемых дашбордов некорректны. Проверьте корректность данных дашбордов перед их импортом.
EVENT_MODEL.IMPORT.INCORRECT_DATA	Данные импортируемых моделей событий некорректны. Убедитесь в правильности данных моделей событий перед импортом.
EVENT_SOURCE.DISABLE.ALREADY_DISABLED	Попытка отключения уже отключенного конвейера. Проверьте статус конвейера перед его отключением.
EVENT_SOURCE.DISABLE.COLLECTOR_DISABLED	Попытка отключения конвейера с отключенным связанным коллектором. Включите связанный коллектор перед отключением конвейера.
EVENT_SOURCE.ENABLE.ALREADY_ENABLED	Попытка включения уже включенного конвейера. Перепроверьте статус конвейера.
EVENT_SOURCE.ENABLE.COLLECTOR_DISABLED	Попытка включения конвейера с отключенным связанным коллектором. Включите коллектор перед включением конвейера.
EVENT_SOURCE.ENABLE.PIPELINE_ERRORS	Попытка включения конвейера с ошибками валидации. Исправьте все ошибки валидации перед включением конвейера.
EVENT_SOURCE.IMPORT.INCORRECT_DATA	Данные импортируемых конвейеров некорректны. Проверьте корректность данных конвейеров перед их импортом.
EVENT_SOURCE.UPDATE.DISABLE_TO_UPDATE	Попытка выполнить операцию для включенного конвейера. Отключите конвейер перед выполнением операции обновления.
EXPERTISE.DELETE.IN_USE	Попытка удаления элемента экспертизы, который используется в одном из конвейеров. Убедитесь, что элемент экспертизы не используется ни на одном из конвейеров перед его удалением.

Код ошибки	Описание
EXPERTISE.ENABLE.NOT_PUBLISHED_YET	Опубликуйте элемент экспертизы, чтобы включить его. Убедитесь, что объект был опубликован перед его включением.
EXPERTISE.PUBLISH.DUPLICATED_ENTITY	Экспертиза с таким ID или с такой версией уже существует. Проверьте и измените ID и/или версию экспертизы перед публикацией.
EXPERTISE.PUBLISH.EMPTY_CONTENT	Не указано содержимое элемента экспертизы. Укажите содержимое элемента экспертизы перед публикацией.
EXPERTISE.PUBLISH.EMPTY_SCHEMA	Не указана схема элемента экспертизы. Укажите схему элемента экспертизы перед публикацией.
EXPERTISE.PUBLISH.EMPTY_URL	Указана пустая ссылка на файл элемента экспертизы. Укажите корректную ссылку на файл элемента экспертизы.
EXPERTISE.PUBLISH.FAIL_DOWNLOAD	Произошла ошибка при загрузке содержимого таблицы обогащения по указанной ссылке. Проверьте доступность и корректность ссылки.
EXPERTISE.PUBLISH.INVALID_SOURCE_CODE	Ошибка в коде элемента экспертизы. Исправьте ошибки в коде элемента экспертизы перед публикацией.
EXPERTISE.PUBLISH.INVALID_VERSION	Укажите корректную версию элемента экспертизы. Убедитесь, что в поле версии указана корректная версия элемента экспертизы.
EXPERTISE.PUBLISH.LOWER_VERSION	Укажите версию больше текущей. Проверьте версию элемента экспертизы и укажите значение больше текущего.
EXPERTISE.PUBLISH.TESTS_FAILED	Во время выполнения тестов произошла ошибка. Исправьте ошибки, выявленные в ходе тестирования, перед публикацией элемента экспертизы.
EXPERTISE.PUBLISH.TRY_TO_CHANGE_ID	Невозможно изменить ID элемента экспертизы. Не пытайтесь изменять ID опубликованного элемента экспертизы.
EXPERTISE.PUBLISH.UNSUPPORTED_LOCATION	Указанный тип содержимого элемента экспертизы не поддерживается. Используйте поддерживаемые типы содержимого для элемента экспертизы.
EXPERTISE.RUN_TESTS.INVALID_SOURCE_CODE	Для запуска тестов необходимо исправить все ошибки в коде. Перед тестированием убедитесь в отсутствии ошибок в коде элемента экспертизы.

Код ошибки	Описание
RESOURCE_FOLDER.DELETE.CONTAINS_RESOURCES	Попытка удаления папки, содержащей ресурсы. Удалите все ресурсы из папки перед ее удалением.
RESOURCE_FOLDER.DELETE.ROOT_FOLDER	Попытка удаления корневой папки. Удаление корневой папки невозможно.
RESOURCE_FOLDER.PATCH.PARENT_FOLDER_AS_CHILD	Попытка присвоения родительской директории ID дочерней. Проверьте корректность структуры директорий.
RESOURCE_FOLDER.PATCH.PARENT_FOLDER_AS_FOLDER	Попытка присвоения родительской директории собственного ID. Убедитесь в корректности изменений в структуре директорий.
RESOURCE_FOLDER.PATCH.PARENT_FOLDER_NOT_EXIST	Попытка присвоения несуществующего ID в качестве родительской директории. Убедитесь в существовании директории с таким ID.
RESOURCE_FOLDER.PATCH.ROOT_FOLDER_CREATING	Попытка создания второго корневого каталога. Создание более одного корневого каталога невозможно.
RESOURCE_FOLDER.PATCH.ROOT_FOLDER_EDITING	Попытка редактирования корневого каталога. Изменение корневого каталога невозможно.

Глоссарий

Управление системой

Модуль

Минимальная функциональная единица системы в рамках платформы R-Vision EVO.

Событие аудита

Внутрисистемное событие, которое создается при выполнении определенных действий, связанных с безопасностью системы. Используется для обеспечения целостности системы.

Подробнее: Аудит системных событий

Управление доступом

LDAP-соединение

Настройка для подключения системы к службам каталогов, таким как Microsoft Active Directory, ALD Pro, FreeIPA, OpenLDAP.

Подробнее: LDAP-соединения

Администратор

Стандартная роль, предназначенная для настройки и поддержки инфраструктуры системы. Обладает всеми возможными разрешениями в системе, имеет полный контроль над всеми тенантами системы.

Подробнее: Стандартные роли

Бот

Учетная запись, предназначенная для автоматизированного взаимодействия с системой через API-запросы.

Подробнее: Пользователи

Группа пользователей

Объединение учетных записей для назначения общих прав управления системой.

Подробнее: Группы пользователей

Доменная группа

Группа пользователей, связанная с каталогом LDAP. Ее состав и данные учетных записей автоматически синхронизируются с указанным LDAP-каталогом.

Подробнее: Группы пользователей

Дочерний тенант

Изолированная часть основной инфраструктуры системы, которая содержит ресурсы и данные, выделенные из основного тенанта. Каждый дочерний тенант работает независимо, обеспечивая доступ и управление только собственными ресурсами и данными.

Подробнее: Тенанты

Инженер

Стандартная роль, предназначенная для настройки операционных процессов системы, разработки решений для оптимизации этих процессов и их внедрения. Действия инженера ограничены рамками определенных тенантов.

Подробнее: Стандартные роли

Локальная группа

Группа пользователей, куда могут быть включены любые учетные записи в системе. Ее состав регулируется вручную через веб-интерфейс системы.

Подробнее: Группы пользователей

Локальный пользователь

Учетная запись для доступа и работы пользователей в системе.

Подробнее: Пользователи

Основной тенант

Тенант, устанавливаемый в системе по умолчанию. Имеет доступ к данным всех дочерних тенантов. Основной тенант нельзя удалить или изменить.

Подробнее: Тенанты

Пользователь

1. Средство идентификации и аутентификации лица в системе, включающее в себя уникальные учетные данные, такие как логин и пароль, а также набор присвоенных ролей.

   Подробнее: Пользователи

2. Стандартная роль, предназначенная для работы с данными системы в рамках определенных тенантов. Имеет ограниченный доступ к ресурсам системы.

   Подробнее: Стандартные роли

Пользователь из домена

Учетная запись, созданная через интеграцию с LDAP-сервером и предназначенная для управления пользователями из внешнего каталога.

Подробнее: Пользователи

Разрешение

Право пользователя на выполнение определенного действия или наличие у него доступа к определенной функциональности системы.

Подробнее: Роли

Роль

Набор разрешений пользователя в системе.

Подробнее: Роли

Стандартная роль

Роль с предопределенным набором разрешений. Стандартную роль нельзя изменить или удалить.

Подробнее: Стандартные роли

Суперадминистратор

Статус пользователя, предоставляющий ему высший уровень доступа к функциональности системы. Суперадминистратор может выполнять любые операции в системе, включая создание и управление другими ролями и пользователями.

Подробнее: Роли

Тенант

Изолированное виртуальное пространство в многопользовательской системе, где данные, настройки и доступы одних частей системы отделены от других.

Подробнее: Тенанты

Управление пространствами

Пространство

Логическая среда в рамках кластера Kubernetes, предназначенная для развертывания и запуска продуктовых сервисов.

Подробнее: Работа с пространствами

Сателлит

Удаленный кластер Kubernetes. Сателлит не обладает полным спектром функциональности системы и может работать только с теми компонентами, которые на нем запущены.

Подробнее: Управление пространствами

Сервис

Программный компонент, который может разворачиваться в продукте. Сервис создается и запускается в рамках пространства.

Подробнее: Работа с сервисами

Центральный кластер

Кластер Kubernetes, в котором разворачиваются все компоненты системы.

Подробнее: Управление пространствами

Управление событиями ИБ

VRL-трансформация

Элемент конвейера, который позволяет обрабатывать и преобразовывать события с применением языка VRL.

Подробнее: Работа с VRL-трансформациями

Агрегатор

Элемент конвейера, который использует правила агрегации для объединения и анализа событий.

Подробнее: Работа с агрегаторами

Активный список

Двумерный массив данных для накопления информации, полученной из определенных полей событий, поступающих на коррелятор.

Подробнее: Работа с активными списками

База данных событий

База данных для хранения событий в системе.

Подробнее: Работа с базами данных событий

Дашборд

Специальная панель, предназначенная для представления информации в графическом виде.

Подробнее: Работа с дашбордами

Коллектор

Компонент системы, который обеспечивает сбор событий из источников, их обработку и дальнейшую отправку в хранилище и/или во внешние системы.

Подробнее: Работа с коллекторами

Конвейер

Компонент коллектора, представляющий собой упорядоченную последовательность взаимосвязанных элементов, выполняющих различные этапы обработки событий.

Подробнее: Работа с конвейерами

Конечная точка

Элемент конвейера, который отвечает за отправку событий.

Подробнее: Работа с конечными точками

Коррелятор

Элемент конвейера, который представляет собой комплекс правил корреляции.

Подробнее: Работа с корреляторами

Маршрутизатор

Элемент конвейера, который позволяет на основе заданных фильтров (маршрутов) направлять

события на разные этапы обработки в зависимости от заданных условий.

Подробнее: Работа с маршрутизаторами

Модель события

Компонент системы, определяющий формат хранилища событий: имена, типы полей и типы данных в полях.

Подробнее: Работа с моделями событий

Нормализатор

Элемент конвейера, который представляет собой комплекс правил нормализации.

Подробнее: Работа с нормализаторами

Обогащение

Компонент коллектора, который позволяет использовать таблицы обогащения в корреляторах и/или нормализаторах конвейеров.

Подробнее: Работа с обогащениями

Отчет

Компонент системы, предназначенный для сбора, анализа и представления данных.

Подробнее: Работа с отчетами

Ресурс

Компонент системы, который обеспечивает настройку ее конкретных функций и управление ими.

Подробнее: *Ресурсы*

Сбор метрик

Элемент конвейера, который собирает метрики различных элементов конвейеров для их дальнейшей обработки и анализа.

Подробнее: Работа со сбором метрик

Секрет

Компонент системы, который обеспечивает безопасное хранение чувствительных данных, таких как учетные данные, токены, строки подключения, сертификаты, приватные ключи.

Подробнее: Работа с секретами

Событие

Идентифицированный переход системы в состояние, которое указывает на возможное нарушение информационной безопасности.

Подробнее: Работа с разделом Поиск

Точка входа

Элемент конвейера, который отвечает за доставку в систему исходных событий.

Подробнее: Работа с точками входа

Универсальная модель события

Системная модель события, предназначенная для хранения информации о событиях в едином, универсальном, виде.

Подробнее: Универсальная модель события

Фильтр

Элемент конвейера, который позволяет сократить объем обрабатываемых событий путем их отсеивания на основе условий на языке VRL.

Подробнее: Работа с фильтрами

Хранилище событий

Таблица в базе данных событий. Используется для хранения событий, поступивших в систему и обработанных коллектором.

Подробнее: Работа с хранилищами событий

Шаблон отчета

Инструмент для определения структуры и периодичности публикации отчета. Позволяет пользователю выбирать аналитические данные для включения в отчет и поддерживает функции автоматического создания и отправки отчета по расписанию.

Подробнее: Работа с отчетами

Шина

Компонент коллектора, который позволяет направлять потоки событий между конвейерами.

Подробнее: Работа с шинами

Управление агентами

WatchDog

Параметр конфигурации группы агентов, который запускает сервис для отслеживания процессов на агенте.

Подробнее: Настройка конфигурации группы агентов

Агент

Бинарный исполняемый файл на конечных устройствах. Агент позволяет производить сбор информации об уязвимостях и угрозах.

Подробнее: Работа с агентами

Группа агентов

Объединение агентов, упрощающее выполнение массовых операций над ними. Для группы доступна настройка конфигурации, которая позволяет задавать политики отслеживания различных действий на ее агентах.

Подробнее: Работа с группами агентов

Инвентарная информация

Агрегированные данные о хосте, собираемые агентом.

Подробнее: Сбор инвентарной информации

Конфигурация группы агентов

Набор правил и настроек, которые регламентируют поведение группы агентов.

Подробнее: Настройка конфигурации группы агентов

Менеджер агентов

Компонент, который разворачивается в центральном кластере и предназначен для управления агентами, группами агентов и шлюзами.

Подробнее: *Агенты*

Шлюз

Промежуточный уровень в сети между агентами, установленными на конечных устройствах, и центральным кластером. Шлюзы предназначены для управления потоками данных между агентами и центральным кластером.

Подробнее: Работа со шлюзами

Управление экспертизой

Глобальная функция

Элемент экспертизы, задающий переиспользуемый блок кода на языке VRL, который можно применять при создании правил корреляции и нормализации.

Подробнее: Глобальные функции

Конструктор правил корреляции

Компонент системы, позволяющий визуально строить правила корреляции в интерактивном режиме без необходимости ручной настройки его RObject-конфигурации.

Подробнее: Построение правила корреляции в конструкторе

Правило агрегации

Элемент экспертизы, который описывает логику объединения и анализа групп событий безопасности.

Подробнее: Правила агрегации

Правило корреляции

Элемент экспертизы, который содержит конфигурацию для автоматизированного распознания определенных последовательностей обрабатываемых событий ИБ.

Подробнее: Правила корреляции

Правило нормализации

Элемент экспертизы, который обрабатывает и преобразовывает поступающие события согласно условиям и коду нормализации на языке VRL.

Подробнее: Правила нормализации

Схема активного списка

Элемент экспертизы, который задает конфигурацию и параметры для создания активного списка и управления им.

Подробнее: Схемы активных списков

Схема дашборда

Элемент экспертизы, который задает конфигурацию и параметры для создания дашборда.

Подробнее: Схемы дашбордов

Схема шаблона отчета

Элемент экспертизы, который задает конфигурацию и параметры для создания шаблона отчета.

Подробнее: Схемы шаблонов отчетов

Таблица обогащения

Элемент экспертизы, который используется для дополнения и уточнения информации по событиям безопасности.

Подробнее: Таблицы обогащения

Шаблон поиска

Элемент экспертизы, который определяет параметры и условия для выполнения запросов к данным в системе в разделе **Поиск** и виджетах дашбордов.

Подробнее: Шаблоны поиска

Экспертиза

Модуль системы, позволяющий создавать объекты на языке RObject для специализированных задач.

Подробнее: Работа с элементами экспертизы

Элемент экспертизы

Объект, создаваемый в разделе **Экспертиза**.

Подробнее: Работа с элементами экспертизы

Моделирование бизнес объектов

Асимметричная связь

Связь, в левой и правой части которой указаны разные сущности.

Подробнее: Структура схемы домена

Атрибут

Характеристика сущности, которая определяется названием, описанием, типом данных значения и дополнительной информацией.

Подробнее: Структура схемы домена

Виджет

Визуальный элемент, позволяющий управлять системой, исходя из текущего контекста взаимодействия с пользователем.

Подробнее: Описания интерфейса

Виджет атрибута

Визуальный элемент, позволяющий отобразить в интерфейсе и редактировать значение атрибута объекта. Используется при формировании представления сущности.

Подробнее: Описания интерфейса

Виджет дашбордов

Виджет, позволяющий обеспечить пользователю доступ в интерфейсе к действиям над сущностями домена.

Подробнее: Описания интерфейса

Виджет действия

Визуальный элемент, позволяющий запустить действие над экземпляром сущности из интерфейса.

Подробнее: Описания интерфейса

Виджет истории изменений

Визуальный элемент, позволяющий отобразить в интерфейсе список событий аудита, связанных с конкретным экземпляром сущности.

Подробнее: Описания интерфейса

Виджет комментариев

Визуальный элемент, позволяющий отобразить в интерфейсе блок комментариев.

Подробнее: Описания интерфейса

Виджет связи

Визуальный элемент, позволяющий отобразить в интерфейсе редактор связи. С помощью редактора можно просматривать текущие экземпляры связи и создавать новые экземпляры связи.

Подробнее: Описания интерфейса

Виджет списка саязей

Визуальный элемент, позволяющий отобразить в интерфейсе редактор связей, с помощью которого можно группировать несколько связей в рамках одного виджета для просмотра и редактирования.

Подробнее: Описания интерфейса

Группа представлений

Контейнер, содержащий другие группы представлений и виджетов. Из групп можно составить более сложные представления, не дублируя элементы схемы домена.

Подробнее: Описания интерфейса

Действие

Событие, связанное с доменом, сущностью или представлением, которое можно вызвать через пользовательский интерфейс или API.

Подробнее: Описания интерфейса

Домен

Логически завершенный модуль платформы, который объединяет данные, бизнес-правила и интерфейс, сохраняя возможность гибкой настройки и интеграции.

Подробнее: Описания интерфейса

Значение атрибута

Значение характеристики, которое имеет экземпляр сущности.

Подробнее: Структура схемы домена

Идентификация

Процесс проверки на уникальность получаемых данных из внешних источников. Настраивается с помощью правил.

Подробнее: Структура схемы домена

Контекстное меню

Меню, которое может размещаться в панели объекта, в панели действий списка и строке списка. К контекстному меню можно привязать только действия.

Подробнее: Описания интерфейса

Линейная связь

Тип связи, позволяющий строить горизонтальные отношения между сущностями, задавая ограничения на количество объектов с левой и правой стороны связи.

Подробнее: Структура схемы домена

Массовое действие

Действие, совершаемое одновременно над несколькими объектами.

Подробнее: Описания интерфейса

Медиаресурсы

Определение иконок в структуре схемы домена.

Подробнее: Описания интерфейса

Меню

Раздел в схеме домена, который определяет возможность размещения элементов меню в основном меню или в меню настроек, обеспечивая переход к представлениям.

Подробнее: Описания интерфейса

Название связи

Отображаемое в интерфейсе название связи, отражающее характер отношения. Задается отдельно для каждой из сторон связи.

Подробнее: Структура схемы домена

Направленная связь

Связь, имеющая строго заданное направление от левой стороны связи к правой, определяя воздействие и объект воздействия. Менять местами сущности связи нельзя.

Подробнее: Структура схемы домена

Наследование

Способ создания новых сущностей путем переопределения свойств существующих сущностей. Унаследованные сущности сохраняют те атрибуты базовых сущностей, которые не были переопределены.

Подробнее: Структура схемы домена

Ненаправленная связь

Связь, не имеющая строго заданного направления от левой стороны связи к правой. Менять местами сущности связи можно.

Подробнее: Структура схемы домена

Полный доменный идентификатор (FQID)

Идентификатор элемента схемы домена, по которому можно ссылаться на него из схемы текущего или другого домена.

Подробнее: Структура схемы домена

Пользовательский тип данных

Тип данных, параметры которого необходимо определить в схеме домена дополнительно.

Подробнее: Типы данных в схемах доменов

Представление

Пользовательский интерфейс для работы с экземплярами сущностей домена.

Подробнее: Описания интерфейса

Представление сущности

Пользовательский интерфейс для просмотра и редактирования экземпляра сущности. Строится из групп виджетов и правил их расстановки.

Подробнее: Описания интерфейса

Рабочий процесс

Тип данных для описания жизненного цикла сущности. Значение рабочего процесса состоит из множества состояний, переходов между ними и требований, выполняемых между переходами.

Подробнее: Типы данных в схемах доменов

Реконсиляция

Процесс сопоставления и согласования данных из разных источников для обеспечения их совместимости и целостности. Настраивается с помощью правил.

Подробнее: Структура схемы домена

Родительская сущность

Сущность, на которую ссылается другая сущность при наследовании.

Подробнее: Структура схемы домена

Связь

Элемент схемы домена, хранящий ссылки на связанные сущности и имеющий собственные характеристики, такие как название, направление, тип связи и симметричность.

Подробнее: Структура схемы домена

Симметричная связь

Связь, в левой и правой части которой указана одна и та же сущность.

Подробнее: Структура схемы домена

Системный атрибут

Атрибут, значение которого заполняется автоматически в момент создания или изменения экземпляра сущности. Значение системного атрибута недоступно для редактирования пользователем, однако доступно для отображения в виджетах и представлениях.

Подробнее: Структура схемы домена

Списочное представление

Пользовательский интерфейс для отображения экземпляров сущности в виде списка с пагинацией, фильтрами и группировками.

Подробнее: Описания интерфейса

Стороны связи

Сущности, между которыми создана связь.

Подробнее: Структура схемы домена

Сущность

Абстракция, описывающая множество однородных бизнес-объектов и их структуру в моделируемой предметной области. Структура включает в себя множество атрибутов и связей.

Подробнее: Структура схемы домена

Сущность потомка

Сущность, которая наследует свойства родителя. Всегда имеет только одного родителя и сама может являться родителем для другого потомка.

Подробнее: Структура схемы домена

Схема домена

Формализованное описание структуры данных и бизнес-правил, на основе которых создается домен.

Подробнее: Схемы доменов

Тип данных

Тип, определяющий множество значений атрибута, его хранение и доступные редакторы.

Подробнее: Типы данных в схемах доменов

Цепочка наследования

Иерархическая последовательность связанных наследованием сущностей.

Подробнее: Структура схемы домена

Экземпляр связи

Отдельный объект, который хранит ссылки на конкретные связанные экземпляры сущностей, а также включает параметры и атрибуты связи.

Подробнее: Структура схемы домена

Экземпляр сущности

Конкретный объект в домене, задаваемый идентификатором и значениями его атрибутов и связей.

Подробнее: Структура схемы домена

Автоматизация бизнес-процессов

Воркер

Сервис системы, выполняющий назначенные ему задачи и использующий для этого выделенные ресурсы системы.

Подробнее: Работа с воркерами

Зависимость

Библиотека, используемая в скриптах и плейбуках.

Подробнее: Плейбуки

Задание

Сущность, включающая в себя идентификатор и версию элемента экспертизы, настройки запуска и значения аргументов для запусков. Одно задание может иметь несколько запусков.

Подробнее: Плейбуки

Запуск

Сущность, создаваемая при однократном выполнении задания. Содержит сведения о статусе, результатах, логах и времени выполнения. Одно задание может иметь несколько запусков.

Подробнее: Работа с запусками

Конфигурация

Объект, задающий конфигурацию внешнего ресурса или системы, с которыми должны взаимодействовать скрипты. Может содержать ссылки на переменные или секреты.

Подробнее: Работа с конфигурациями

Переменная

Именованные данные вида "ключ-значение", которые можно переиспользовать в разных ресурсах системы, в том числе в плейбуках. Переменная может быть засекречена.

Подробнее: Работа с переменными

Плейбук

Набор связанных сценариев, представленных в виде ориентированного ациклического графа. Этот граф задает последовательность выполнения, ветвления, циклы, повторы и точки принятия решения. Плейбуки позволяют задавать и исполнять сложные процессы обработки, оркестрируя базовые сценарии.

Подробнее: Плейбуки

Резолюция

Решение, принятое пользователем по запросу, поступившему на согласование в ходе работы плейбука. Может быть положительным или отрицательным.

Подробнее: Работа с резолюциями

Скрипт

Программа для решения конкретной задачи.

Подробнее: Скрипты

Схема конфигурации

Набор данных, который используется для задания структуры конфигурации.

Подробнее: Схемы конфигурации

Триггер

Сущность, которая связывает событие и запуск, а также вызывает запуск плейбука или скрипта.

Подробнее: Работа с триггерами

Управление сканированием

Агентское сканирование

Процесс сканирования оборудования, на котором установлены агенты, и получения информации об этом оборудовании. Осуществляется модулем управления агентами.

Подробнее: Агентское сканирование

Базовое сканирование

Тип профиля сканирования, позволяющий сканировать сеть с целью обнаружения активных устройств и сбора базовой инвентарной информации.

Подробнее: Профили сканирования

Безопасная проверка

Скрипт профиля **Тестирование на проникновение**, проверяющий сведения информативного характера.

Подробнее: Вкладки профилей

Небезопасная проверка

Скрипт профиля **Тестирование на проникновение**, который может причинить вред оборудованию при эксплуатации.

Подробнее: Вкладки профилей

Обнаружение

Тип профиля сканирования, позволяющий сканировать сеть с целью обнаружения активных устройств и открытых портов.

Подробнее: Профили сканирования

Подбор паролей

Процесс подбора паролей для получения несанкционированного доступа к системам и учетным записям или зашифрованным данным. Проводится для выявления слабых мест в аутентификации.

Подробнее: Вкладки профилей

Проверка стандартов

Тип профиля сканирования, позволяющий сканировать сеть с целью проверки соответствия системы установленным нормативным требованиям и стандартам безопасности.

Подробнее: Профили сканирования

Тестирование на проникновение

Тип профиля сканирования, позволяющий сканировать сеть с целью обнаружения активных сервисов и поиска уязвимостей с помощью безопасных и небезопасных проверок.

Подробнее: Профили сканирования

Управление уязвимостями

База данных уязвимостей

Структурированное хранилище информации об известных уязвимостях на устройствах. База данных содержит описания, классификацию, оценки рисков и рекомендации по устранению уязвимостей.

Подробнее: Обновление базы уязвимостей

Проверка соответствия

Запись с информацией, полученной при запуске задачи сканирования на основе профиля типа **Проверка стандартов**. Содержит сведения о проверке соответствия системы установленным нормативным требованиям и стандартам безопасности.

Подробнее: Проверки

Уязвимость

Сущность в системе, которая содержит собранные сканерами или добавленные вручную сведения об уязвимости.

Подробнее: Уязвимости