

# ???????????? KICS for Networks

- [API KICS for Networks](#)
- [Настройка передачи событий из KICS for Networks \(версий 4.3 и 4.5\) в KUMA](#)

# API KICS for Networks

## ???????????????? Kaspersky Industrial CyberSecurity for Networks API

В Kaspersky Industrial CyberSecurity for Networks реализован интерфейс прикладного программирования, который обеспечивает доступ к функциям программы для сторонних приложений.

В комплект поставки Kaspersky Industrial CyberSecurity for Networks входит пакет с описаниями спецификаций для представления данных в запросах к серверу REST API. Сервер REST API функционирует на компьютере Сервера Kaspersky Industrial CyberSecurity for Networks и обрабатывает запросы с использованием архитектурного стиля взаимодействия REST. Обращения к серверу REST API выполняются по протоколу HTTPS. Вы можете настроить параметры сервера REST API в разделе Параметры → Серверы подключений (в том числе заменить используемый по умолчанию самоподписанный сертификат на доверенный).

Для представления данных в запросах и ответах используется формат JSON.

Документация с описанием запросов на основе архитектурного стиля REST публикуется в виде онлайн-справки на странице [Kaspersky Online Help](#).

1. [ОТКРЫТЬ ДОКУМЕНТАЦИЮ С ОПИСАНИЕМ ЗАПРОСОВ К СЕРВЕРУ REST API, версия 3](#)
2. [ОТКРЫТЬ ДОКУМЕНТАЦИЮ С ОПИСАНИЕМ ЗАПРОСОВ К СЕРВЕРУ REST API, версия 4](#)

## ???????? ???? ?????????????????

Рассмотрим пример использования API для локального экспорта данных об устройствах из KICS for Networks.

Для начала нужно создать коннектор, с типом Generic и именем, например, **GetDeviceAll**, указать пароль и адрес сервера, узел размещения коннектора в нашем случае будет такой же, как и адрес сервера (Рисунок 1). После создания коннектора у вас скачается файл свертки, который будет называться как коннектор\_GetDeviceAll.zip.

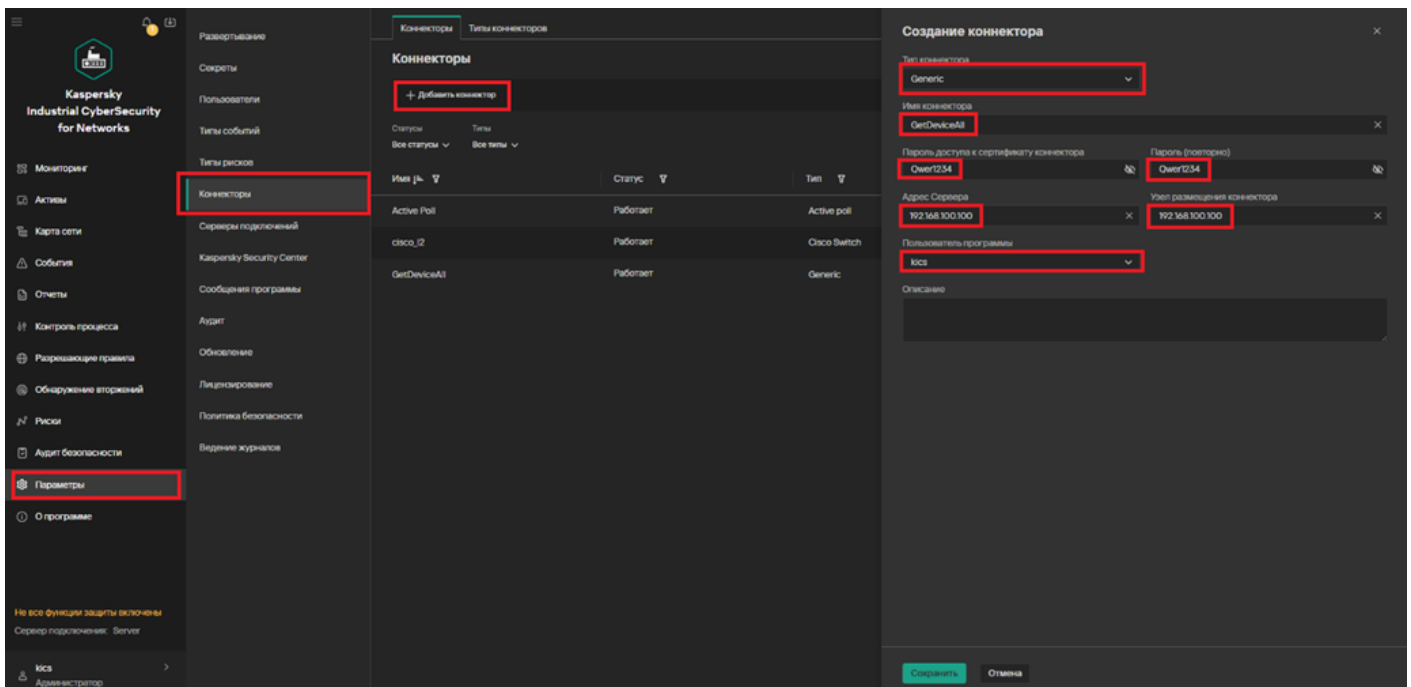


Рисунок 1 – Создание коннектора

Далее необходимо создать директории для корректной работы кода. Впишите команды ниже последовательно:

```
cd

mkdir GetDeviceAll && cd GetDeviceAll

mkdir connector && cd connector && cp /opt/kaspersky/kics4net-
connectors/libexec/default_connectors_registrar.py default_connectors_registrar.py
```

Далее в папку connector необходимо положить файл свертки коннектор\_GetDeviceAll.zip, который мы получили при создании коннектора и распаковать его в эту директорию командой:

```
unzip -d коннектор_GetDeviceAll коннектор_GetDeviceAll.zip
```

Далее необходимо зарегистрировать коннектор командой:

```
sudo python3 default_connectors_registrar.py create
```

Далее необходимо выполнить действия показанные на Рисунке 2: ввести имя коннектора GetDeviceAll, ввести полный путь до файла свертки коннектор\_GetDeviceAll.zip (/home/<username>/GetDeviceAll/connector/коннектор\_GetDeviceAll.zip), ввести пароль, который вы указывали при создании коннектора в веб-интерфейсе и согласиться с запуском службы коннектора GetDeviceAll.

```
kics@k4net-server:~/GetDeviceAll/connector$ sudo python3 default_connectors_registrar.py create
2025-07-18 17:14:19 INFO ===== KICS for Networks 4.3 =====
2025-07-18 17:14:19 INFO ===== Connector registrar =====
Connector name:
GetDeviceAll
Registration package path:
/home/kics/GetDeviceAll/connector/коннектор_GetDeviceAll.zip
Certificate access password:
2025-07-18 17:15:50 INFO Ensure the certificate store directory exists: /var/opt/kaspersky/kics4net-connectors/instances/GetDeviceAll/ssl
2025-07-18 17:15:50 INFO Unpacking registration pack: /home/kics/GetDeviceAll/connector/коннектор_GetDeviceAll.zip
2025-07-18 17:15:50 INFO Extracting webserver.pem
2025-07-18 17:15:50 INFO Extracting credentials
2025-07-18 17:15:51 INFO Parsing metadata
['stem', 'syslog', 'email', 'generic', 'kuma', 'cisco_switch'] Generic
2025-07-18 17:15:51 INFO Registration pack metadata:
2025-07-18 17:15:51 INFO Name: 192.168.100.100
2025-07-18 17:15:51 INFO ID: 4
2025-07-18 17:15:51 INFO Server URI: https://192.168.100.100:8080/kics/api
2025-07-18 17:15:51 INFO Type: Generic
2025-07-18 17:15:51 INFO Storing the certificate
2025-07-18 17:15:51 INFO Storing the private key
2025-07-18 17:15:51 INFO Service: kics4net-connectors@GetDeviceAll has been created
Do you want to start service? y/n
y
Created symlink /etc/systemd/system/multi-user.target.wants/kics4net-connectors@GetDeviceAll.service -> /lib/systemd/system/kics4net-connectors@.s
2025-07-18 17:15:54 INFO Service GetDeviceAll was enabled
2025-07-18 17:15:54 INFO Service GetDeviceAll was started
```

Рисунок 2 – Регистрация коннектора

После этого статус коннектора в KICS for Networks поменяется с “Ожидает регистрации” на “Работает”

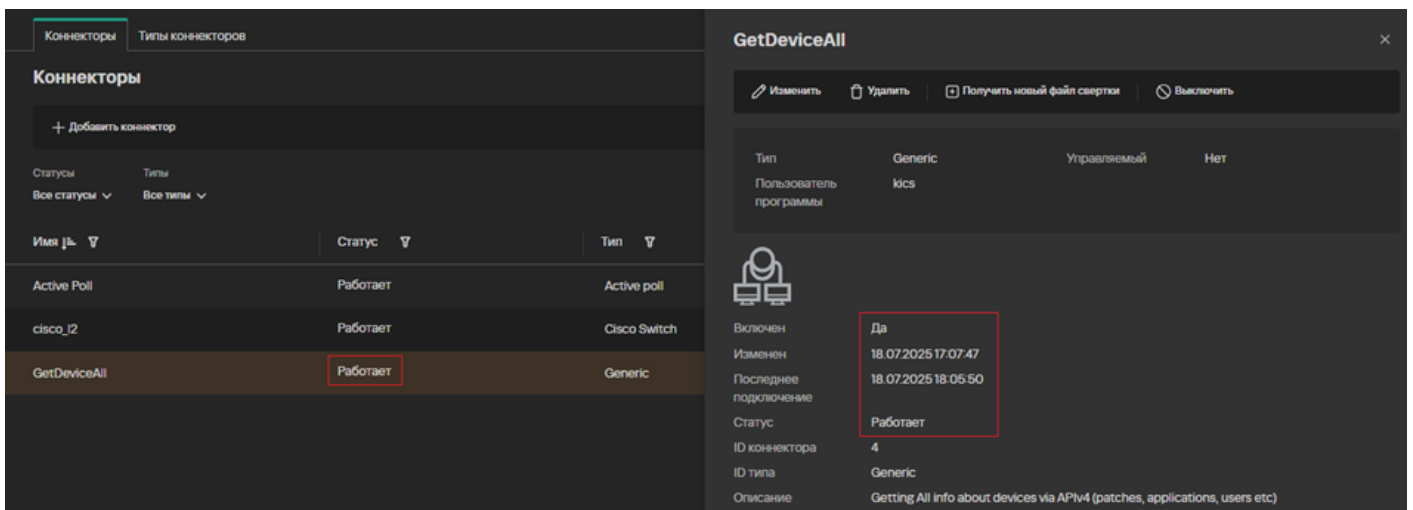


Рисунок 3 – Статус работы коннектора в Веб-консоли

Теперь можно перейти к написанию кода, который будет взаимодействовать с KICS for Networks по API.

Код будет писаться на Python, но вы можете использовать любой удобный для вас язык

Для начала напишем класс подключения к API сервера:

```
import logging
import json
import requests
from requests.exceptions import RequestException
```

```
from time import sleep

class KicsApiClient:
    def __init__(self, metadata_path, cert_path, key_path):
        self.metadata_path = metadata_path
        self.cert_path = cert_path
        self.key_path = key_path
        self.base_url = None
        self.token = None
        self._initialize()

    def _initialize(self):
        with open(self.metadata_path, 'r', encoding='utf-8') as f:
            metadata = json.load(f)
        self.base_url = metadata.get('serverUri', '').rstrip('/')
        if not self.base_url.endswith('/kics/api'):
            self.base_url += '/kics/api'
        logging.info(f"Базовый URL API: {self.base_url}")
        self._authenticate()

    def _authenticate(self):
        url = f"{self.base_url}/auth/token?grant_type=certificate"
        resp = requests.post(
            url,
            cert=(self.cert_path, self.key_path),
            verify=False
        )
        resp.raise_for_status()
        self.token = resp.json().get('access_token')
        if not self.token:
            raise ValueError("Аутентификация не удалась: токен не получен")
        logging.info("Аутентификация успешна")

    def _post_query(self, path, payload, offset=0, limit=1000, delay=1):
        all_items = []
        headers = {
            'Authorization': f'Bearer {self.token}',
            'Content-Type': 'application/json'
        }
        }
```

```

while True:
    body = payload.copy()
    body.update({'offset': offset, 'limit': limit})
    url = f"{self.base_url}{path}"
    try:
        resp = requests.post(url, headers=headers, json=body, verify=False)
        resp.raise_for_status()
    except RequestException as e:
        logging.error(f"Ошибка запроса к {url}: {e}")
        if hasattr(e, 'response') and e.response is not None:
            try:
                logging.error(f"Текст ответа: {e.response.text}")
            except:
                pass
        raise
    try:
        json_resp = resp.json()
    except ValueError:
        break
    data = json_resp.get('values') or json_resp.get('elements')
    if not data:
        break
    all_items.extend(data)
    logging.info(f"Получено {len(data)} записей из {path}, всего: {len(all_items)}")
    if len(data) < limit:
        break
    offset += limit
    sleep(delay)
return all_items

```

Теперь добавим в этот класс функции которые будут вызываться в классе экспорта данных:

```

def get_device_applications(self):
    return self._post_query('/v4/device-apps/query', {})

def get_device_patches(self):
    return self._post_query('/v4/device-patches/query', {})

def get_device_users(self):
    return self._post_query('/v4/device-users/query', {})

```

Перейдем к написанию класса, который будет отвечать за экспорт данных из KICS for Networks по API и будет конвертировать полученные данные в csv файл:

```
import csv

class CsvExporter:
    @staticmethod
    def export_dicts(items, filename):
        if not items:
            logging.warning(f"Нет данных для файла {filename}")
            return

        fieldnames = set()
        for item in items:
            fieldnames.update(item.keys())
        fieldnames = sorted(fieldnames)
        headers_rus = [RUSSIAN_HEADERS.get(fn, fn) for fn in fieldnames]
        with open(filename, 'w', newline='', encoding='utf-8-sig') as f:
            writer = csv.writer(f, delimiter=';')
            writer.writerow(headers_rus)
            for item in items:
                row = [item.get(fn, '') for fn in fieldnames]
                writer.writerow(row)
        logging.info(f"Экспортировано {len(items)} записей в {filename}")
```

В этом классе используется глобальная переменная `RUSSIAN_HEADERS`, которая хранит в себе ключ - значение, для перевода ключей на русский язык:

```
RUSSIAN_HEADERS = {
    'Id': 'Идентификатор',
    'id': 'Идентификатор',
    'DeviceId': 'Идентификатор устройства',
    'device': 'Устройство',
    'ApplicationName': 'Название приложения',
    'name': 'Имя',
    'fullName': 'Полное имя',
    'ApplicationVersion': 'Версия приложения',
    'PatchVersion': 'Версия патча',
    'UserName': 'Имя пользователя',
    'description': 'Описание',
```

```
'Domain': 'Домен',
'groups': 'Группы',
'isDisabled': 'Отключен',
'isLockedOut': 'Заблокирован',
'lastSeen': 'Последний вход',
'canChangePassword': 'Можно сменить пароль',
'mustChangePassword': 'Требуется смена пароля',
'passwordNeverExpires': 'Пароль не истекает',
'sid': 'SID',
'source': 'Источник',
'installedOn': 'Установлено на',
'version': 'Версия',
'vendor': 'Производитель',
'program': 'Программа',
'size': 'Размер',
}
```

Почти все готово! Осталось написать main, который объединит в себе весь функционал нашего кода:

```
import os
import sys

def main():
    BASE_DIR = os.path.dirname(os.path.abspath(__file__))
    META = os.path.join(BASE_DIR, 'connector', 'коннектор_GetDeviceAll', 'metadata.json')
    CERT = '/var/opt/kaspersky/kics4net-connectors/instances/GetDeviceAll/ssl/connector.crt'
    KEY = '/var/opt/kaspersky/kics4net-connectors/instances/GetDeviceAll/ssl/connector.key'
    for path in (META, CERT, KEY):
        if not os.path.exists(path):
            logging.error(f"Не найден файл: {path}")
            sys.exit(1)
    client = KicsApiClient(META, CERT, KEY)
    apps = client.get_device_applications()
    patches = client.get_device_patches()
    users = client.get_device_users()
    CsvExporter.export_dicts(apps, 'DeviceApplications.csv')
    CsvExporter.export_dicts(patches, 'DevicePatches.csv')
    CsvExporter.export_dicts(users, 'DeviceUsers.csv')
```

```
if __name__ == '__main__':  
    main()
```

Отлично! Код готов, теперь необходимо перенести файл GetDeviceAll.py в папку GetDeviceAll любым известным вам способом. В конечном счете директория должна выглядеть вот так:

```
└─ ./home/<username>  
  └─ ./GetDeviceAll  
    └─ ./connector  
      └─ ./коннектор_GetDeviceAll.zip  
        └─ ./default_connectors_registrar.py  
          └─ ./коннектор_GetDeviceAll  
            └─ ./certificates.pfx  
              └─ ./metadata.json  
                └─ ./webserver.pem  
                  └─ ./GetDeviceAll.py
```

Далее в консоли перейдем в директорию GetDeviceAll командой:

```
cd /home/<username>
```

И запустить файл GetDeviceAll.py командой:

```
sudo python3 GetDeviceAll.py
```

В итоге мы получили 3 файла DeviceApplications.csv, DevicePatches.csv, DeviceUsers.csv которые находятся в этой же директории GetDeviceAll.

```
kics@k4net-server:~/GetDeviceAll$ sudo python3 GetDeviceAll.py  
2025-07-18 18:05:50 INFO Базовый URL API: https://192.168.100.100:8080/kics/api  
2025-07-18 18:05:50 INFO Аутентификация успешна  
2025-07-18 18:05:50 INFO Получено 31 записей из /v4/device-apps/query, всего: 31  
2025-07-18 18:05:50 INFO Получено 9 записей из /v4/device-patches/query, всего: 9  
2025-07-18 18:05:50 INFO Получено 16 записей из /v4/device-users/query, всего: 16  
2025-07-18 18:05:50 INFO Экспортировано 31 записей в DeviceApplications.csv  
2025-07-18 18:05:50 INFO Экспортировано 9 записей в DevicePatches.csv  
2025-07-18 18:05:50 INFO Экспортировано 16 записей в DeviceUsers.csv
```

Рисунок 4 – Вывод логов скрипта по экспорту данных



Рисунок 5 – Результат работы скрипта

## P.S.

Полный код :)

```
import os
import sys
import json
import logging
import requests
import csv

from requests.exceptions import RequestException
import urllib3
from time import sleep

urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

logging.basicConfig(
    format='%(asctime)s %(levelname)s %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S',
    level=logging.INFO
)

RUSSIAN_HEADERS = {
    'Id': 'Идентификатор',
    'id': 'Идентификатор',
    'DeviceId': 'Идентификатор устройства',
    'device': 'Устройство',
    'ApplicationName': 'Название приложения',
```

```
'name': 'Имя',
'fullName': 'Полное имя',
'ApplicationVersion': 'Версия приложения',
'PatchVersion': 'Версия патча',
'UserName': 'Имя пользователя',
'description': 'Описание',
'Domain': 'Домен',
'groups': 'Группы',
'isDisabled': 'Отключен',
'isLockedOut': 'Заблокирован',
'lastSeen': 'Последний вход',
'canChangePassword': 'Можно сменить пароль',
'mustChangePassword': 'Требуется смена пароля',
'passwordNeverExpires': 'Пароль не истекает',
'sid': 'SID',
'source': 'Источник',
'installedOn': 'Установлено на',
'version': 'Версия',
'vendor': 'Производитель',
'program': 'Программа',
'size': 'Размер',
}
```

```
class KicsApiClient:
    def __init__(self, metadata_path, cert_path, key_path):
        self.metadata_path = metadata_path
        self.cert_path = cert_path
        self.key_path = key_path
        self.base_url = None
        self.token = None
        self._initialize()

    def _initialize(self):
        with open(self.metadata_path, 'r', encoding='utf-8') as f:
            metadata = json.load(f)
        self.base_url = metadata.get('serverUri', '').rstrip('/')
        if not self.base_url.endswith('/kics/api'):
            self.base_url += '/kics/api'
        logging.info(f"Базовый URL API: {self.base_url}")
        self._authenticate()
```

```

def _authenticate(self):
    url = f"{self.base_url}/auth/token?grant_type=certificate"
    resp = requests.post(
        url,
        cert=(self.cert_path, self.key_path),
        verify=False
    )
    resp.raise_for_status()
    self.token = resp.json().get('access_token')
    if not self.token:
        raise ValueError("Аутентификация не удалась: токен не получен")
    logging.info("Аутентификация успешна")

def _post_query(self, path, payload, offset=0, limit=1000, delay=1):
    all_items = []
    headers = {
        'Authorization': f'Bearer {self.token}',
        'Content-Type': 'application/json'
    }
    while True:
        body = payload.copy()
        body.update({'offset': offset, 'limit': limit})
        url = f"{self.base_url}{path}"
        try:
            resp = requests.post(url, headers=headers, json=body, verify=False)
            resp.raise_for_status()
        except RequestException as e:
            logging.error(f"Ошибка запроса к {url}: {e}")
            if hasattr(e, 'response') and e.response is not None:
                try:
                    logging.error(f"Текст ответа: {e.response.text}")
                except:
                    pass
            raise
        try:
            json_resp = resp.json()
        except ValueError:
            break
        data = json_resp.get('values') or json_resp.get('elements')

```

```
        if not data:
            break
        all_items.extend(data)
        logging.info(f"Получено {len(data)} записей из {path}, всего: {len(all_items)}")
        if len(data) < limit:
            break
        offset += limit
        sleep(delay)
    return all_items
```

```
def get_device_applications(self):
    return self._post_query('/v4/device-apps/query', {})
```

```
def get_device_patches(self):
    return self._post_query('/v4/device-patches/query', {})
```

```
def get_device_users(self):
    return self._post_query('/v4/device-users/query', {})
```

```
class CsvExporter:
```

```
    @staticmethod
```

```
    def export_dicts(items, filename):
```

```
        if not items:
```

```
            logging.warning(f"Нет данных для файла {filename}")
```

```
            return
```

```
        fieldnames = set()
```

```
        for item in items:
```

```
            fieldnames.update(item.keys())
```

```
        fieldnames = sorted(fieldnames)
```

```
        headers_rus = [RUSSIAN_HEADERS.get(fn, fn) for fn in fieldnames]
```

```
        with open(filename, 'w', newline='', encoding='utf-8-sig') as f:
```

```
            writer = csv.writer(f, delimiter=';')
```

```
            writer.writerow(headers_rus)
```

```
            for item in items:
```

```
                row = [item.get(fn, '') for fn in fieldnames]
```

```
                writer.writerow(row)
```

```
        logging.info(f"Экспортировано {len(items)} записей в {filename}")
```

```
def main():
```

```
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
META = os.path.join(BASE_DIR, 'connector', 'коннектор_GetDeviceAll', 'metadata.json')
CERT = '/var/opt/kaspersky/kics4net-connectors/instances/GetDeviceAll/ssl/connector.crt'
KEY = '/var/opt/kaspersky/kics4net-connectors/instances/GetDeviceAll/ssl/connector.key'
for path in (META, CERT, KEY):
    if not os.path.exists(path):
        logging.error(f"Не найден файл: {path}")
        sys.exit(1)
client = KicsApiClient(META, CERT, KEY)
apps = client.get_device_applications()
patches = client.get_device_patches()
users = client.get_device_users()
CsvExporter.export_dicts(apps, 'DeviceApplications.csv')
CsvExporter.export_dicts(patches, 'DevicePatches.csv')
CsvExporter.export_dicts(users, 'DeviceUsers.csv')

if __name__ == '__main__':
    main()
```

# ?????????? ?????????? ??????????

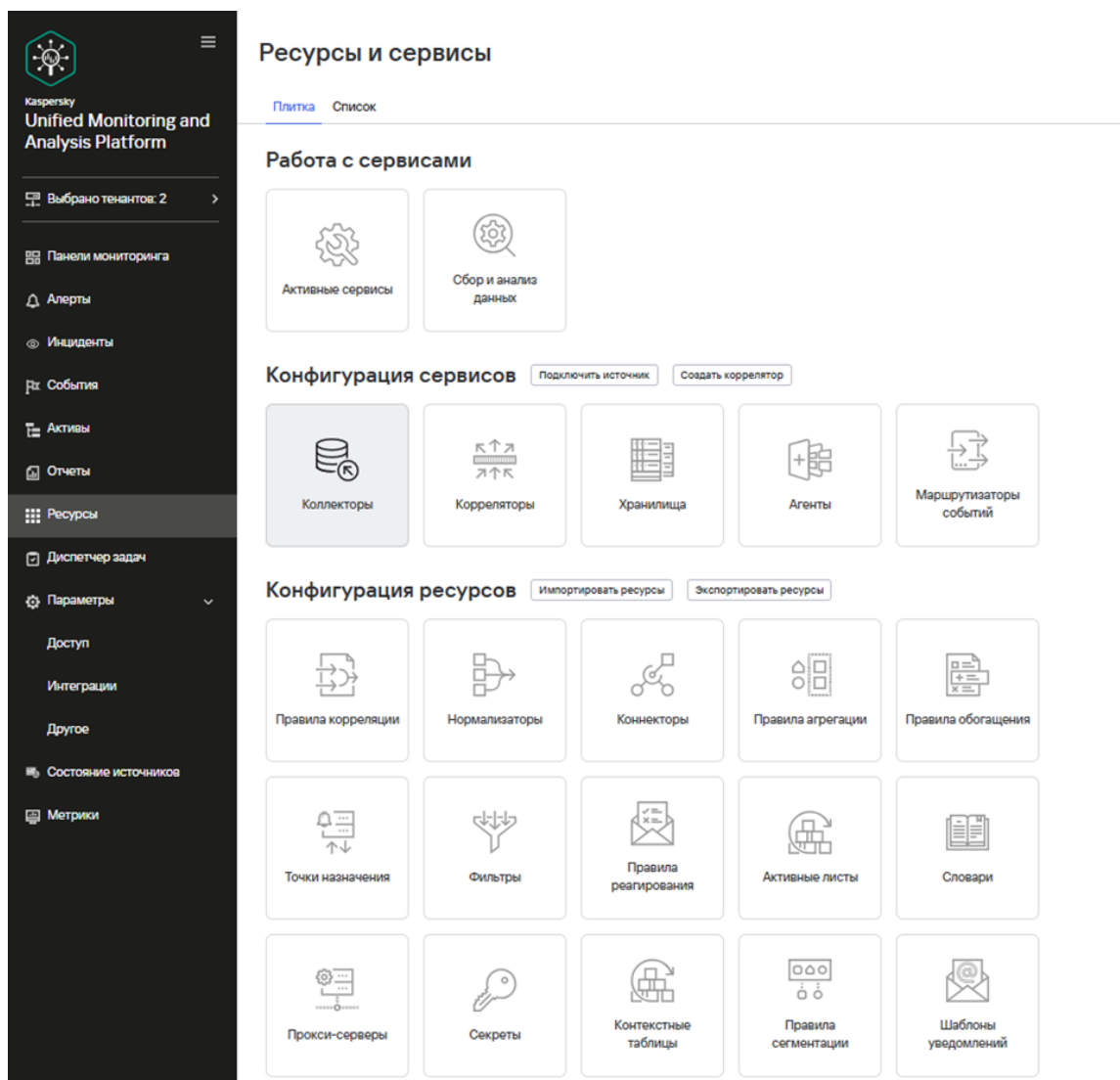
## ?? KICS for Networks (????????

### 4.3 ? 4.5) ? KUMA

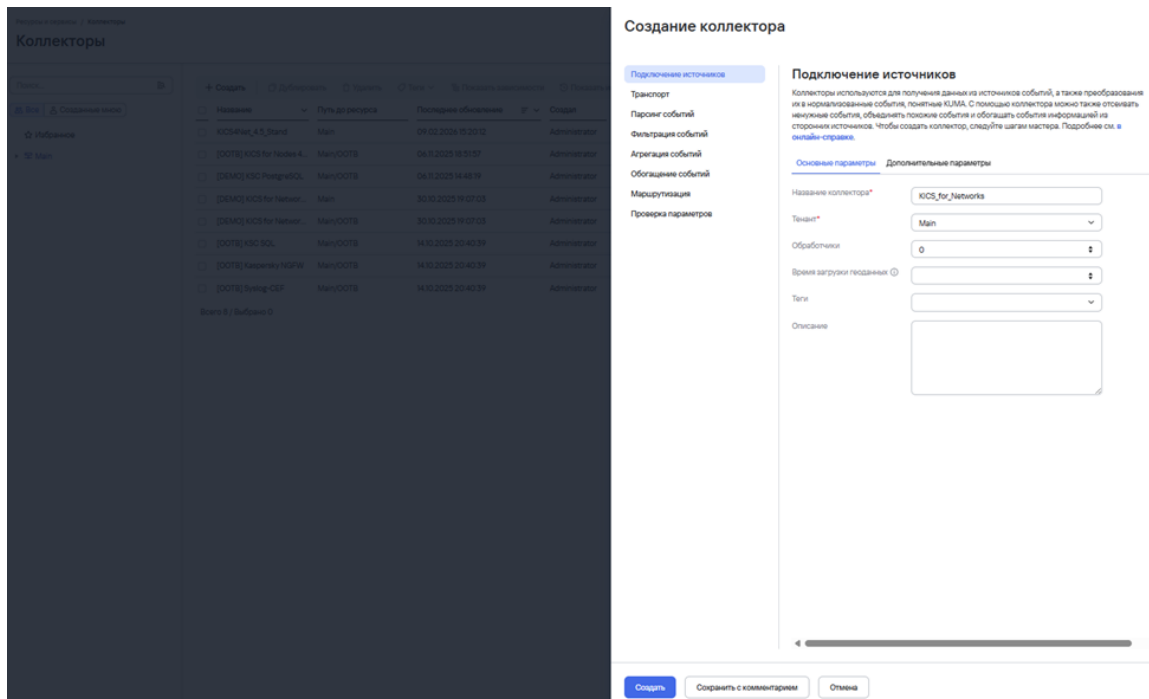
Настройка интеграции выполняется в два этапа: сначала создается и настраивается коллектор в KUMA, затем создается коннектор в KICS for Networks.

#### Настройка KUMA

1. Перейдите в раздел «Ресурсы» и откройте вкладку «Коллекторы».



1. Нажмите кнопку «+ Создать».



1. В открывшемся окне укажите название коллектора. Рекомендуется использовать наименование продукта, тип подключения и порт (например, KICS\_for\_Networks\_TCP(5160)), чтобы в дальнейшем легко идентифицировать коллектор.

## Создание коллектора

### Подключение источников

- Транспорт
- Парсинг событий
- Фильтрация событий
- Агрегация событий
- Обогащение событий
- Маршрутизация
- Проверка параметров

### Подключение источников

Коллекторы используются для получения данных из источников событий, а также преобразования их в нормализованные события, понятные KUMA. С помощью коллектора можно также отсеивать ненужные события, объединять похожие события и обогащать события информацией из сторонних источников. Чтобы создать коллектор, следуйте шагам мастера. Подробнее см. в [онлайн-справке](#).

[Основные параметры](#) [Дополнительные параметры](#)

Название коллектора*	<input type="text" value="KICS_for_Networks_TCP/5160"/>
Тенант*	<input type="text" value="Main"/>
Обработчики	<input type="text" value="0"/>
Время загрузки геоданных ⓘ	<input type="text"/>
Теги	<input type="text"/>
Описание	<input type="text"/>

1. На вкладке «Транспорт» укажите тип соединения (протокол) и порт, который будет использоваться для приема событий от источника.

## Создание коллектора

### Подключение источников

#### Транспорт

- Транспорт
- Парсинг событий
- Фильтрация событий
- Агрегация событий
- Обогащение событий
- Маршрутизация
- Проверка параметров

### Транспорт

Подключите источник, от которого хотите получать события. Подробнее см. в [онлайн-справке](#).

[Основные параметры](#) [Дополнительные параметры](#)

Коннектор	<input type="text" value="Создать"/>
Тип* ⓘ	<input type="text" value="tcp"/>
URL* ⓘ	<input type="text" value=":5160"/>
Auditd	<input type="checkbox"/>
Разделитель	<input type="text"/>

1. На вкладке «Парсинг событий» - «Настройки парсинга» выберите соответствующий нормализатор для обработки входящих событий.

## Создание коллектора



Подключение источников

Транспорт

Парсинг событий

Фильтрация событий

Агрегация событий

Обогащение событий

Маршрутизация

Проверка параметров

### Парсинг событий

Преобразуйте полученные события в формат, понятный KUMA. Подробнее см. [в онлайн-справке](#).  
Настройки парсинга доступны, если тип коннектора — http/tcp/udp.

Схемы парсинга [Настройки парсинга](#)

IP-адрес(-а)

Введите IP-адрес или несколько IP-адресов, используя запятую в качестве разделителя

Нормализатор\*

+ Добавить условную нормализацию

+ Источник события

1. На вкладке «Маршрутизация» добавьте ранее созданные коррелятор и хранилище. Подробнее о разворачивании этих компонентов см. в [Справке по KUMA](#).

## Создание коллектора



Подключение источников

Транспорт

Парсинг событий

Фильтрация событий

Агрегация событий

Обогащение событий

Маршрутизация

Проверка параметров

### Маршрутизация

Укажите, куда следует отправлять полученные события. Подробнее см. [в онлайн-справке](#).

+ Добавить    Удалить

<input type="checkbox"/>	Название	Тип	URL
<input type="checkbox"/>	[OOTB] Correlator	correlator	kuma-ics.demo.lab:7231
<input type="checkbox"/>	[OOTB] Storage	storage	kuma-ics.demo.lab:7230

1. На вкладке «Проверка параметров» нажмите кнопку «Сохранить и создать сервис».

## Создание коллектора

Подключение источников

Транспорт

Парсинг событий

Фильтрация событий

Агрегация событий

Обогащение событий

Маршрутизация

Проверка параметров

### Проверка параметров

Настройка коллектора завершена, сервис добавлен в KUMA. Подробнее см. [в онлайн-справке](#).

Чтобы начать получать события, сервис этого коллектора необходимо установить на сервере, предназначенном для сбора событий (см. пример команды установки ниже). Обратите внимание, что должна быть обеспечена сетевая связность компонентов системы и открыты порты.

Подробнее см. [в онлайн-справке](#).

Сохранить и создать сервис

1. После создания коллектора скопируйте сгенерированную команду и выполните её в терминале сервера KUMA с правами администратора для развертывания сервиса.

Подключение источников

Транспорт

Парсинг событий

Фильтрация событий

Агрегация событий

Обогащение событий

Маршрутизация

Проверка параметров

### Проверка параметров

Настройка коллектора завершена, сервис добавлен в KUMA. Подробнее см. [в онлайн-справке](#).

Чтобы начать получать события, сервис этого коллектора необходимо установить на сервере, предназначенном для сбора событий (см. пример команды установки ниже). Обратите внимание, что должна быть обеспечена сетевая связность компонентов системы и открыты порты. Подробнее см. [в онлайн-справке](#).

### Сервисы, использующие этот коллектор

Тип	Название
коллектор	KICS_for_Networks_TCP/5160

Сохранить и перезапустить сервисы

Сохранить и обновить параметры сервисов

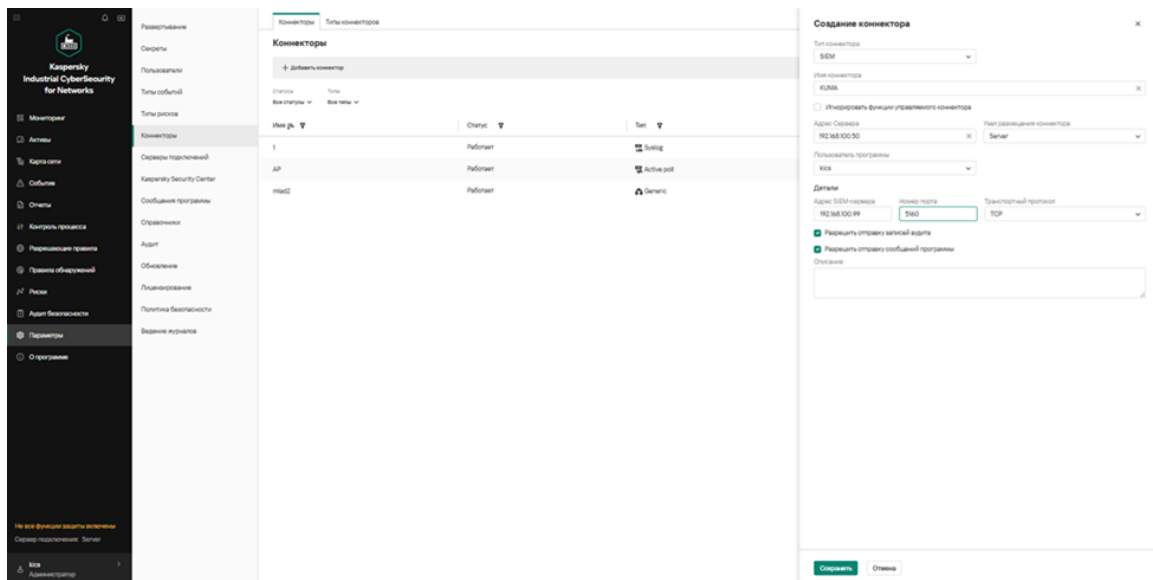
### Рекомендуемая команда для установки коллектора

```
/opt/kaspersky/kuma/kuma collector --core https://kuma-ics.demo.lab:7210 --id 4800ba10-8c0f-4a18-ad48-aad7b446ebc6 --api.port 7237 --install
```






## Настройка KICS for Networks

1. Перейдите в раздел «Параметры» - «Коннекторы» и нажмите кнопку «+ Добавить коннектор».
2. В карточке нового коннектора заполните поля:
  - Тип коннектора: SIEM.
  - Имя коннектора: Произвольное наименование.
  - Адрес сервера: IP-адрес узла, на котором развернут сервер KICS for Networks.
  - Узел размещения коннектора: Server.
  - Пользователь программы: Учетная запись администратора KICS for Networks.
  - Адрес SIEM-сервера: IP-адрес узла, на котором развернут сервер KUMA.
  - Номер порта: Порт, указанный при настройке коллектора KUMA (см. п. 4 раздела «Настройка KUMA»).
  - Транспортный протокол: Протокол, указанный при настройке коллектора KUMA (см. п. 4 раздела «Настройка KUMA»).



1. Нажмите кнопку «Сохранить».
2. Убедитесь, что в свойствах созданного коннектора параметры имеют следующие значения:
  - Включен: Да.
  - Статус: Работает.

 Изменить  Удалить  Выключить

Тип	SIEM	Управляемый	Да
Пользователь программы	kics	Узел размещения коннектора	Server



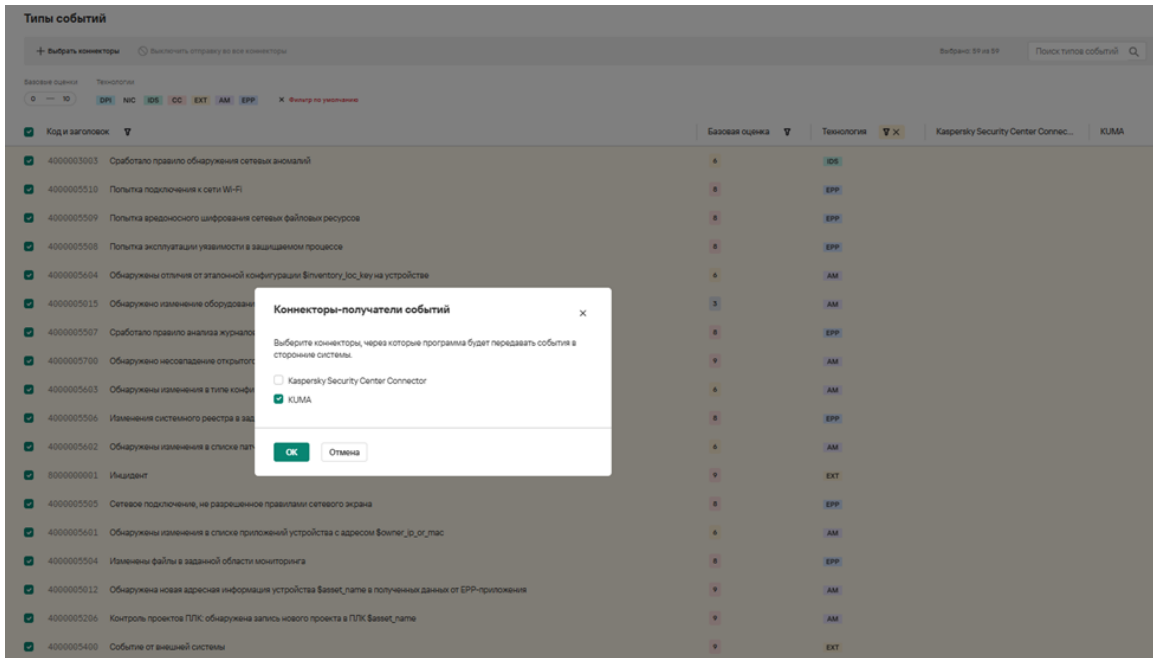
Включен	Да
Изменен	16.02.2026 12:36:45
Последнее подключение	16.02.2026 12:36:45
Статус	Работает
ID коннектора	11
ID типа	Siem
Типы событий	Не отправляются
Функциональные возможности	Отправка данных, Управляемый

#### Детали

Адрес SIEM-сервера	192.168.100.99
Номер порта	5160
Транспортный протокол	TCP
Разрешить отправку записей аудита	
Разрешить отправку сообщений программы	

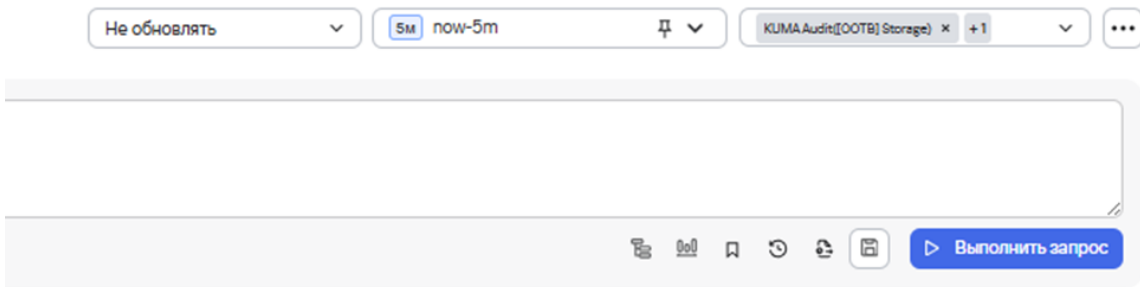
Если значения отличаются, проверьте выполненные настройки и перезапустите коннектор (выполните последовательность «Выключить» - «Включить»).

3. Перейдите в раздел «Параметры» - «Типы событий». Выберите типы событий, которые необходимо передавать в KUMA, и активируйте для них отправку через созданный коннектор.

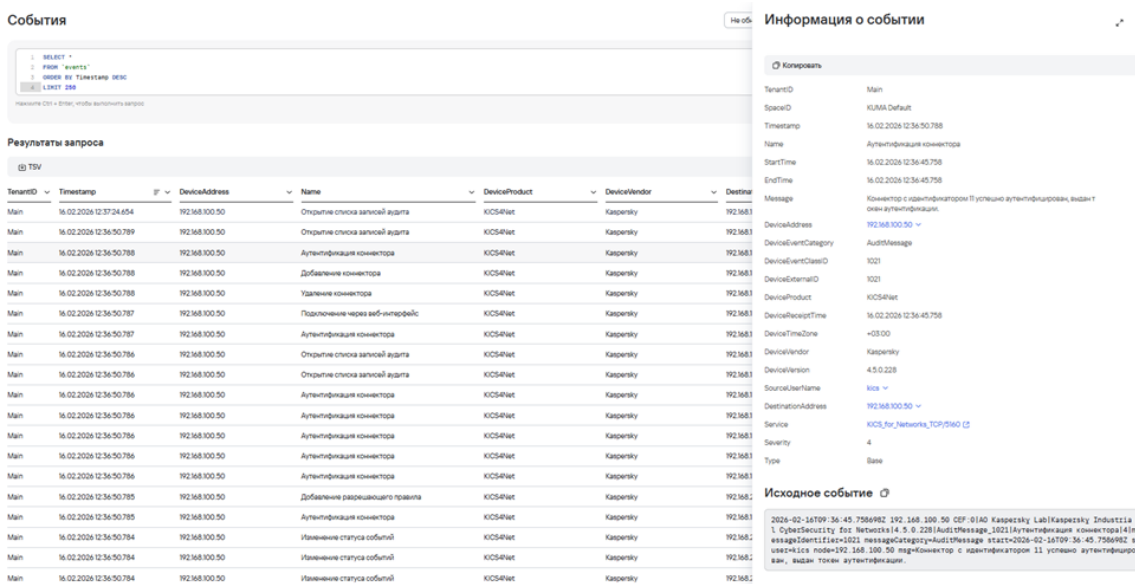


## Проверка передачи событий

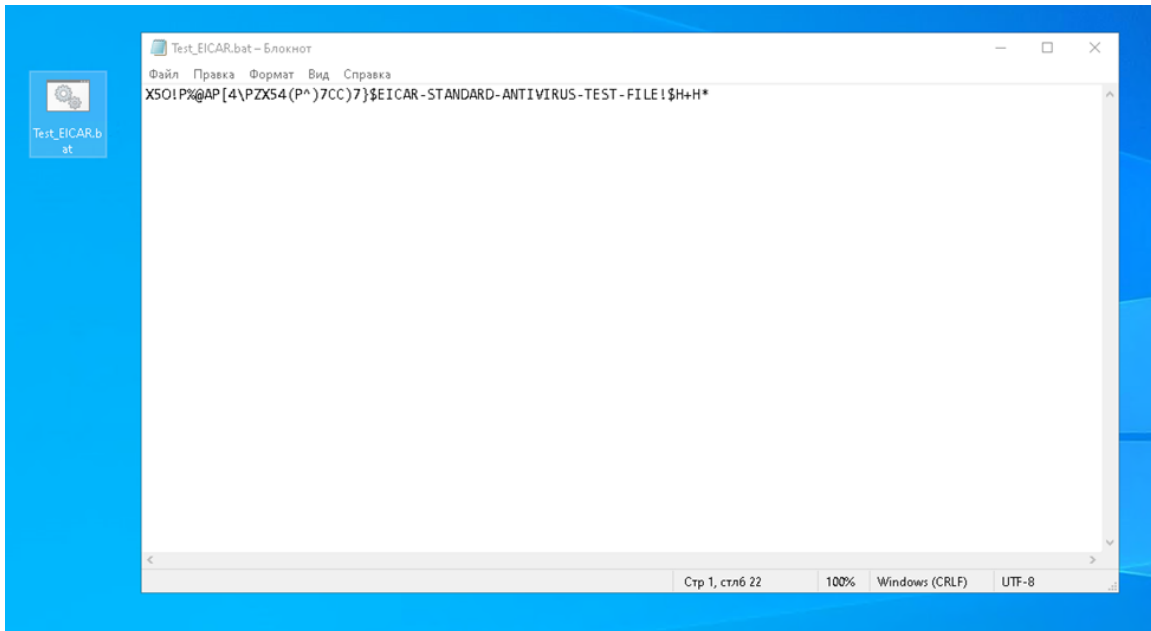
1. В интерфейсе KUMA перейдите в раздел «События» и выполните опрос источников.



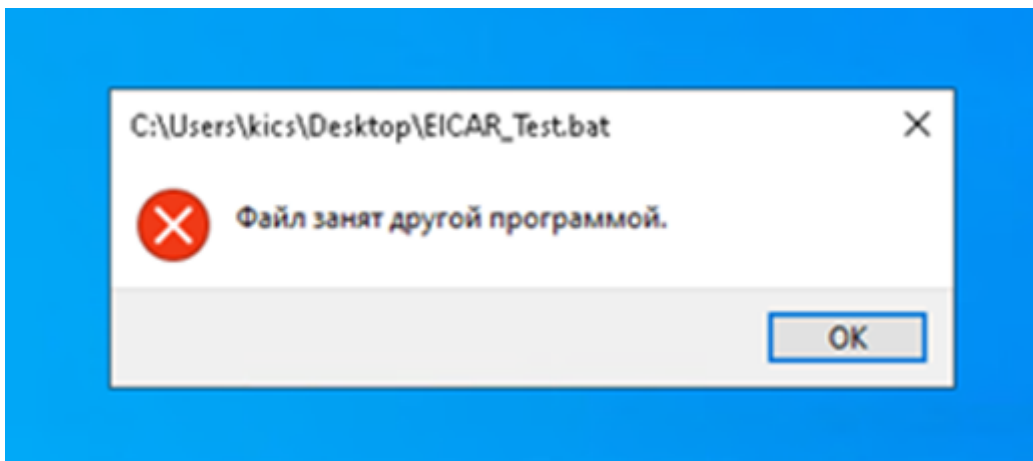
2. Убедитесь в наличии событие об успешной аутентификации коннектора, которое должно появиться в момент его создания в KICS for Networks.



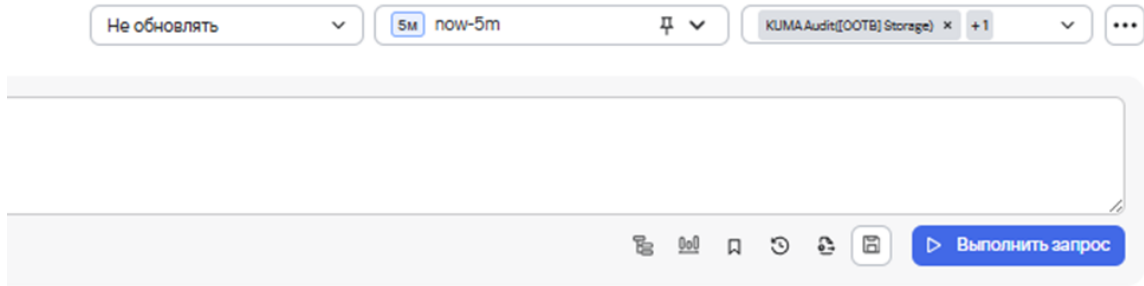
1. Воспроизведите событие безопасности, например, запустите тестовый вирус на узле, который интегрирован с KICS for Networks.



1. Убедитесь, что сработал модуль постоянной защиты KICS for Nodes (детектирование события на стороне источника).



1. В интерфейсе KUMA снова перейдите в раздел «События» и выполните опрос.



1. Убедитесь, что воспроизведенное событие безопасности успешно доставлено и отображается в KUMA.

### События

```

SELECT *
FROM 'events'
ORDER BY Timestamp DESC
LIMIT 200
        
```

Нажмите Ctrl + Enter, чтобы выполнить запрос

### Результаты запроса

TSV

TenantID	Timestamp	IP	DeviceAddress	Name	DeviceProduct	DeviceVendor	Destinat
Main	16.02.2026 12:58:42.147		192.168.100.50	Обнаружено несоответствие (TIME CORRECTION T...	KICS4Net	Kaspersky	
Main	16.02.2026 12:56:10.017		192.168.100.50	Привлечены активности вредоносной программы на у...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:10.016		192.168.100.50	C:\Users\kcor\Desktop\Тест_EICAR bat	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:39.992		192.168.100.50	С IP-адреса 192.168.100.50 часть попыток несрав...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:33.566		192.168.100.50	Обнаружено несоответствие (TIME CORRECTION T...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:33.566		192.168.100.50	Попытки сетевых взаимодействий с IP-адресом 92...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:33.585		192.168.100.50	Обнаружено повреждение (CLOCK_ACCURACY DECRE...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:33.585		192.168.100.50	Обнаружено несоответствие (TIME CORRECTION T...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:27.230		192.168.100.50	Попытки сетевых взаимодействий с IP-адресом 92...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:27.230		192.168.100.50	Попытки сетевых взаимодействий с IP-адресом 92...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:27.230		192.168.100.50	Обнаружено несоответствие (TIME CORRECTION T...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:27.229		192.168.100.50	Обнаружено несоответствие (TIME CORRECTION T...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:23.968		192.168.100.50	Подозрительная активность (MITRE: Unauthorized C...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:23.967		192.168.100.50	Попытки сетевых взаимодействий с IP-адресом 90.18...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:23.967		192.168.100.50	Попытки сетевых взаимодействий с IP-адресом 90.1...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:23.967		192.168.100.50	Обнаружено несоответствие (TIME CORRECTION T...	KICS4Net	Kaspersky	
Main	16.02.2026 12:54:23.966		192.168.100.50	Обнаружено несоответствие (TIME CORRECTION T...	KICS4Net	Kaspersky	
Main	16.02.2026 12:52:35.138		192.168.100.50	С IP-адреса 192.168.100.50 часть попыток несрав...	KICS4Net	Kaspersky	
Main	16.02.2026 12:52:16.425		192.168.100.50	Попытки сетевых взаимодействий с IP-адресом 92...	KICS4Net	Kaspersky	
Main	16.02.2026 12:52:16.425		192.168.100.50	Попытки сетевых взаимодействий с IP-адресом 92...	KICS4Net	Kaspersky	
Main	16.02.2026 12:52:16.425		192.168.100.50	Обнаружено несоответствие (TIME CORRECTION T...	KICS4Net	Kaspersky	

### Информация о событии

Копировать

TenantID	Main
SpaceID	KUMA Default
Timestamp	16.02.2026 12:56:10.017
Name	Привлечены активности вредоносной программы на ус...
StartTime	16.02.2026 12:55:48.222
EndTime	16.02.2026 12:55:48.222
DeviceAddress	192.168.100.50
DeviceEventCategory	Event
DeviceEventClassID	8000000001
DeviceProduct	KICS4Net
DeviceReceiptTime	16.02.2026 12:55:48.222
DeviceTimeZone	+03:00
DeviceVendor	Kaspersky
DeviceVersion	4.5.0.228
SourceHostName	dekltop-4725ka
DeviceCustomString1	K4N_L4_3Win10
DeviceCustomStringLabel	assetName
DeviceCustomString2	None
DeviceCustomStringLabel	monitoringPoint
DeviceCustomString5	VMware201
DeviceCustomStringLabel	arcVendor
DeviceCustomString6	1
DeviceCustomStringLabel	type
Service	KICS_for_Networks_TCP/5160_C3
ApplicationProtocol	IP
BaseEventCount	1
EventOutcome	trojan_on_host
ExternalID	45719
FileString1	Привлечены активности вредоносной программы на ус...
FileStringLabel	technologyRule