

AlertX AI

SECURITY OPERATIONS PLATFORM

Интеграция по API Двустороннее взаимодействие

Как внешняя система — ITSM, SOAR, тикет-система или скрипты автоматизации — интегрируется с AlertX в обе стороны: входящий канал через REST API и приём событий, исходящий — через webhook с подписью HMAC, каналы уведомлений и WebSocket.

REST API

WEBHOOK

HMAC-SHA256

WEBSOCKET

ITSM

SOAR

МУЛЬТИТЕНАНТНОСТЬ

ВЕРСИЯ ДОКУМЕНТА

1.0

ДАТА

17 июля 2026

СТАТУС

Актуальная редакция

КЛАССИФИКАЦИЯ

Внутренний документ

Содержание

1	Аутентификация
1.1	Выпуск API-ключа
1.2	Ограничения
2	Входящая интеграция: внешняя система → AlertX
2.1	REST API (управление объектами)
2.2	Загрузка лог-файлов через API
2.3	Потоковый приём событий (не-REST)
3	Исходящая интеграция: AlertX → внешняя система
3.1	Webhook (основной канал для интеграций)
3.2	Каналы уведомлений (готовые интеграции)
3.3	WebSocket (реальное время)
4	Эталонный двусторонний сценарий (ITSM)
5	Требования безопасности к интеграции

О документе

Документ описывает двустороннюю интеграцию внешних систем с AlertX. Базовый URL API: <https://<host>/api> — TLS обязателен, терминатция на Nginx. Полный перечень эндпоинтов вынесен в отдельный справочник [api-reference.md](#).

3	11	4	2
СПОСОБА АУТЕНТИФИКАЦИИ	КАНАЛОВ УВЕДОМЛЕНИЙ	КАНАЛА ПРИЁМА СОБЫТИЙ	НАПРАВЛЕНИЯ ОБМЕНА

▪ Модель взаимодействия

Внешняя система обменивается с AlertX по двум независимым каналам. Входящий канал используется для управления объектами и передачи событий, исходящий — для доставки алертов наружу. Каналы настраиваются и живут отдельно: отказ одного не блокирует другой.

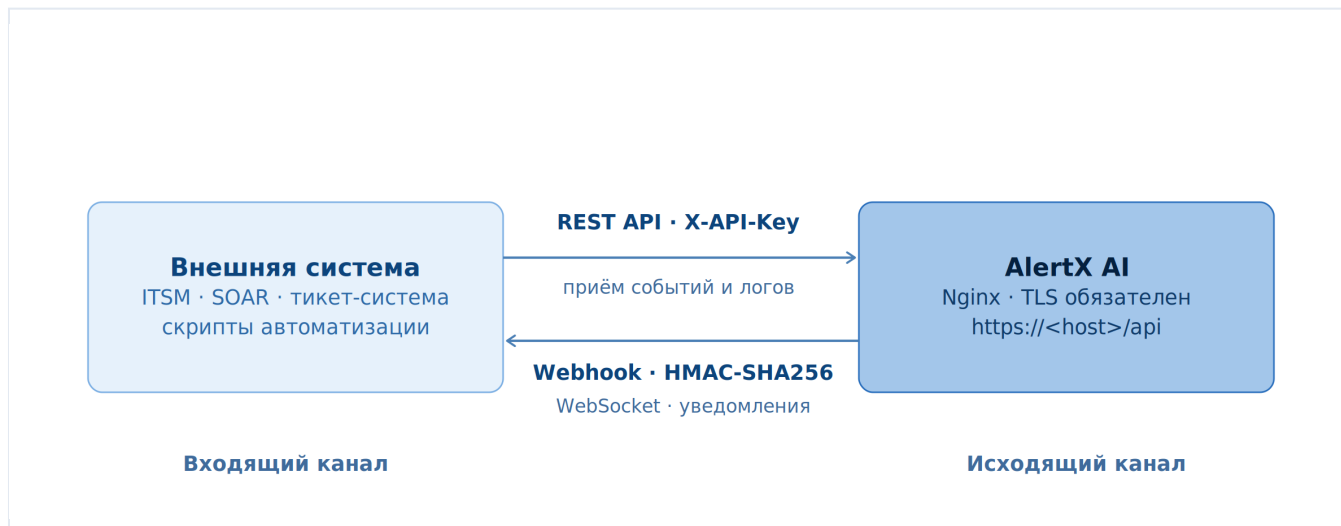


Схема 1. Двустороннее взаимодействие: входящий канал — REST API с ключом [X-API-Key](#) и приём событий; исходящий — webhook с подписью HMAC, WebSocket и каналы уведомлений.

НАПРАВЛЕНИЕ	МЕХАНИЗМЫ	РАЗДЕЛ
Внешняя система → AlertX	REST API, загрузка файлов, Syslog / Beats / Wazuh / Kafka	Раздел 2
AlertX → внешняя система	Webhook (HMAC-SHA256), 11 каналов уведомлений, WebSocket	Раздел 3

1 Аутентификация

Три способа, проверяются в следующем порядке:

СПОСОБ	ЗАГОЛОВОК / МЕХАНИЗМ	НАЗНАЧЕНИЕ
API-ключ	X-API-Key: <key>	Автоматизация, скрипты, межсистемная интеграция. Рекомендуемый способ для интеграций.
Bearer JWT	Authorization: Bearer <token>	API-клиенты; токен получается через POST /api/auth/login
HttpOnly cookie	access_token	Только браузер (веб-интерфейс)

1.1 Выпуск API-ключа

```
POST /api/auth/api-keys
Content-Type: application/json
Authorization: Bearer <token> (или cookie активной сессии)

{ "name": "itsm-integration" }
```

Ответ содержит ключ **один раз** — сохранить сразу, повторно не показывается.

Управление: [GET /api/auth/api-keys](#) (список), [DELETE /api/auth/api-keys/{key_hash}](#) (отзыв). Ключ наследует роль и [org_id](#) создавшего пользователя — все запросы автоматически ограничены его организацией.

Хранение ключей — 24 ч в кэше Redis + персистентно в OpenSearch (шифрование через ConfigStore).

1.2 Ограничения

- Rate limit на логин: 5/мин с одного IP; на загрузку логов: 20/мин.
- 5 неудачных попыток входа → блокировка аккаунта на 15 минут.
- Мультитенантность: все ответы фильтруются по [org_id](#) ключа; [super_admin](#) видит все организации.

2 Входящая интеграция: внешняя система → AlertX

2.1 REST API (управление объектами)

Типовые операции для интеграции (все — с заголовком [X-API-Key](#)):

ОПЕРАЦИЯ	ЭНДПОИНТ
Список алертов (фильтры: severity, статус, время, IP)	GET /api/alerts
Смена статуса алерта (ack / in_progress / closed)	эндпоинты Alert Actions
Создание/чтение/обновление инцидента	POST /api/soar/incidents GET /api/soar/incidents PATCH /api/soar/incidents
Запуск плейбука	POST /api/soar/playbooks/{id}/run
Поиск по логам (Lucene / SQL / PPL)	POST /api/discover/search , /api/sql
Threat Intelligence (индикаторы)	эндпоинты /api/intel/*

Точные схемы запросов/ответов — [api-reference.md](#).

2.2 Загрузка лог-файлов через API

```
POST /api/ingest/upload
X-API-Key: <key>
Content-Type: multipart/form-data

file=<файл>, log_type=<тип>, dataset=<опционально>,
org_id=<опционально, только super_admin>
```

Ответ `202 Accepted` + квитанция. Файл уходит в тот же конвейер парсинга Logstash, что и логи коллекторов. Требуется роль администратора. История: `GET /api/ingest/uploads`.

2.3 Поточковый приём событий (не-REST)

Для непрерывной отправки событий внешними источниками:

КАНАЛ	ПОРТ / ПРОТОКОЛ	ФОРМАТ
Syslog	514/UDP	syslog / CEF
Beats	5044/TCP	Lumberjack (Filebeat, Winlogbeat, Metricbeat)
Wazuh	5000	Wazuh events
Kafka	9093/TCP, mutual TLS	topic-based, MirrorMaker (удалённые коллекторы)

Рекомендация. Разовые выгрузки — через `POST /api/ingest/upload` ; постоянный поток — через `syslog/Beats`; удалённые площадки — через коллектор (буферизация при обрывах связи, см. продуктовую спецификацию).

3 Исходящая интеграция: AlertX → внешняя система

3.1 Webhook (основной канал для интеграций)

Настраивается рег-организация: URL, секрет, минимальный severity (`webhook_min_severity` — алерты ниже порога не отправляются). При каждом прошедшем фильтр алерте AlertX делает `POST` на указанный URL.

Заголовки запроса

```
Content-Type: application/json
User-Agent: AlertX-AI/4.2
X-AlertX-OrgID: <org_id>
X-AlertX-AlertID: <alert_id>
X-AlertX-Signature: sha256=<hex>      (если задан секрет)
```

Тело (событие alert)

```
{
  "event": "alert",
  "org_id": "default",
  "timestamp": "2026-07-17T12:00:00+00:00",
  "alert": {
    "id": "...",
    "name": "MySQL Unauthorized Access Attempt",
    "severity": 8,
    "severity_label": "critical",
    "severity_override": null,
    "src_ip": "192.168.0.100",
    "dst_ip": "...",
    "user": "...",
    "signature_id": "...",
    "device_vendor": "...",
    "device_product": "...",
    "message": "...",
    "ai_summary": "...",
    "ai_risk_level": "...",
    "status": "new",
    "timestamp": "2026-07-17T11:59:58+00:00"
  }
}
```

Тестовое событие (кнопка «Тест» в UI или API) приходит с `"event": "test"` — приёмник обязан отвечать на него `2xx`, **не создавая тикетов**.

Подпись HMAC-SHA256

Если в настройках задан секрет, подпись считается от сырого тела запроса. Проверка на стороне приёмника:

```
import hmac, hashlib

def verify(raw_body: bytes, header: str, secret: str) -> bool:
    expected = "sha256=" + hmac.new(
        secret.encode(), raw_body, hashlib.sha256
    ).hexdigest()
    return hmac.compare_digest(expected, header)
```

Запросы без валидной подписи приёмник должен отбрасывать.

Модель доставки и ограничения

- Доставка fire-and-forget: один POST с таймаутом, **без автоматических повторов**. Ответ `4xx/5xx` логируется на стороне AlertX. Приёмник должен отвечать быстро (`2xx` сразу, обработка асинхронно).
- **URL webhook обязан быть внешним**: адреса частных диапазонов (RFC 1918, loopback, link-local) блокируются SSRF-защитой, включая проверку резолва DNS на момент отправки (защита от DNS rebinding). Приёмник внутри периметра — публиковать через reverse-proxy с публичным именем.
- Гарантированная доставка критична → дублировать канал: webhook + опрос `GET /api/alerts` по расписанию (реконсильация пропущенного).

3.2 Каналы уведомлений (готовые интеграции)

11 каналов, настраиваются на странице «Уведомления»: Email, Slack, Telegram, Teams, PagerDuty, OpsGenie, **Jira**, **ServiceNow**, **TheHive**, Google Chat, Webhook. Для тикет-систем (Jira/ServiceNow/TheHive) AlertX сам создаёт тикеты по алертам — если нужна только регистрация тикетов, кастомный webhook-приёмник не обязателен.

3.3 WebSocket (реальное время)

`wss://<host>/api/ws/alerts` — поток новых алертов в реальном времени (тот же канал, что питает страницу «Алерты»). Статус подключений: `GET /api/ws/status`. Подходит для дашбордов и live-консолей внешних SOC.

4 Эталонный двусторонний сценарий (ITSM)

1. **AlertX → ITSM**: алерт severity $\geq$ порога → webhook POST → ITSM проверяет `X-AlertX-Signature`, создаёт тикет, в тикете сохраняет `alert.id`.
2. ITSM отвечает `200` немедленно (обработка — в фоне).
3. **ITSM → AlertX**: аналитик закрывает тикет → ITSM вызывает API AlertX с `X-API-Key`: смена статуса алерта на `closed` / комментарий / обновление инцидента.
4. Реконсильация: раз в N минут ITSM опрашивает `GET /api/alerts?status=new&...` — подхватывает алерты, чей webhook не дошёл.
5. Тестирование связки: кнопка «Тест» канала Webhook (`event: "test"`) + `POST /api/auth/api-keys` для выпуска отдельного ключа под интеграцию.



Схема 2. Эталонный цикл ITSM: от срабатывания алерта до закрытия тикета и обратной смены статуса в AlertX; реконсильация опросом закрывает пропуски webhook.

5 Требования безопасности к интеграции

- Только HTTPS/WSS; ключи и секреты **не передаются в query-string** — только в заголовках.
- Отдельный API-ключ на каждую интеграцию (именованный) — отзыв без влияния на остальные; ротация не реже раза в квартал.
- Секрет webhook задавать обязательно; приёмник валидирует подпись через constant-time сравнение ([hmac.compare_digest](#)).
- Ограничить приём webhook по IP AlertX-сервера (allowlist) в дополнение к подписи.
- Учётка, от которой выпущен ключ, должна иметь минимально необходимую роль; для чисто читающих интеграций — не админскую.
- Все действия по API попадают в аудит-лог AlertX (страница «Настройки» → аудит).