База знаний и интеграции wiSLA 6

FAQ и практические материалы wiSLA 6: типовые проблемы, диагностика, проверенные решения, рецепты мониторинга, примеры интеграций и шаблонов.

- Интеграции
  - Grafana
  - Уведомления
  - Воркфлоу уведомлений
  - Профили каналов
  - Веб-сценарий vs availability

Интеграции

Интеграции

Grafana

Grafana: интеграция и provisioning

wiSLA подключает внешний Grafana как **слой визуализации** (аналог embed/datasource у wiSLA). Отдельный официальный datasource-плагин — фаза 2; сейчас продукт создаёт **managed datasources** напрямую через Grafana HTTP API.

Настройки

Настройки → Интеграции → Grafana — см. Настройки (админ).

Ключ	Смысл
`grafana.enabled`	Интеграция активна
`grafana.url`	Базовый URL Grafana
`grafana.api_token`	Service account token (Search + provisioning)
`grafana.embed_enabled`	Переключатель wiSLA | Grafana на `/dashboard`
`grafana.default_source`	`wisla` | `grafana` без query-параметров
`grafana.default_dashboard_uid`	UID дашборда по умолчанию
`grafana.catalog_scope`	`wisla` (папка **wiSLA 6** / тег `wisla6`) или `all`
`grafana.auto_provision`	Авто-provisioning при сохранении интеграции

Managed datasources

При provisioning backend читает параметры wiSLA и создаёт в Grafana:

Имя	UID	Источник в wiSLA
wiSLA ClickHouse	`wisla-ch`	`wisla.clickhouse.*`
wiSLA PostgreSQL	`wisla-pg`	`spring.datasource.*`

Требуется плагин **grafana-clickhouse-datasource** на стороне Grafana.

Provisioning дашборда

Автоматически: при «Сохранить и активировать» Grafana, если `auto_provision=true`.

Вручную: кнопка **Provision дашборда** в карточке интеграции или:

```
POST /api/v1/grafana/provision
```

Создаётся:

- папка **wiSLA 6** (`uid=wisla6`);
- дашборд **Обзор инфраструктуры (wiSLA 6)** (`uid=wisla-alpha-ops`);
- JSON шаблон: `backend/src/main/resources/grafana/wisla-infra-overview.json`.

Если `default_dashboard_uid` пуст — после первого provisioning выставляется `wisla-alpha-ops`. Источник на `/dashboard` остаётся **wiSLA**, пока админ не выберет Grafana в интеграции или звёздочкой «по умолчанию».

Ops-скрипт (альфа / legacy)

provision-alpha-grafana-wisla-dashboard.py — для стенда с SSH к Grafana. Синхронизация JSON в backend:

```
python docs/ops/provision-alpha-grafana-wisla-dashboard.py --dump-json > backend/src/main/resources/grafana/wisla-infra-overview.json
```

(на Windows — через `python -c` и запись в файл в UTF-8, см. комментарий в скрипте).

API (кроме settings)

Метод	Путь	Роли
GET	/api/v1/grafana/status	authenticated
GET	/api/v1/grafana/dashboards?q=&scope=wisla|all	authenticated
GET	/api/v1/grafana/embed-url?uid=&theme=	authenticated
PUT	/api/v1/grafana/defaults	ADMIN
POST	/api/v1/grafana/provision	ADMIN

Страница Дашборд

См. Дашборд (пользователь):

- переключатель wiSLA | Grafana;
- звезда «по умолчанию» (ADMIN);
- Share с query-параметрами;
- localStorage `wisla.dashboard.lastView` (источник, uid, id wiSLA).

Grafana server

- `allow_embedding = true`
- CSP `frame-ancestors` на origin wiSLA
- для iframe без второго логина — SSO или anonymous org viewer

Уведомления

Уведомления

Как устроена доставка алертов в wiSLA: **где открывать в UI**, **что делает каждая роль**, **куда смотреть в коде**.

Маршрутизация «кому и когда» — **воркфлоу**. Личные адреса — **профиль пользователя**. Транспорт (SMTP / боты) — **Интеграции**.

Цепочка (логика продукта)

```
Problem (событие)
  → внутренний fault 1:1
  → domain event (fault.opened | taken | reassigned | cleared)
  → матчер активных опубликованных воркфлоу
  → блоки Notify / Ticket
  → транспорт (SMTP / Telegram / MAX / webhook / Jira / Tracker)
```

Kill-switch `notifications.enabled=false` глушит **адаптеры** (и HTTP-тикеты воркфлоу), даже если сценарии активны.

Справочник по блокам и envelope: Воркфлоу уведомлений (guide).

Холст для оператора: Воркфлоу.

Где что в UI

Место	URL	Кто	Что
Настройки → Уведомления	`/settings?section=notifications`	ADMIN, INFOSEC (read)	Отдельные блоки: **звук в браузере**, **push-toast**, kill-switch и ссылки на воркфлоу / интеграции / профили
Администрирование → Воркфлоу	`/workflows`	все роли с доступом к холсту	Список сценариев; клик → редактор
Редактор воркфлоу	`/workflows/{id}`	ADMIN, OPERATOR (+ VIEWER read)	Холст, публикация, запуски, шаринг
Настройки → Интеграции	`/settings?section=integrations`	ADMIN (+ INFOSEC read)	SMTP, Telegram, MAX, webhook, SIEM, **Jira / Tracker для Тикета**; кнопка **Тест**
Аккаунт → Уведомления	`/account` (вкладка)	любой	Свои email / Telegram / MAX; мин. критичность и тихие часы; **Проверить** канал; звук проблем

Место	URL	Кто	Что
Аудит и доступ → Пользователи	`/access/users`	ADMIN пишет, INFOSEC читает	Кнопка **Уведомления** на строке — те же личные каналы за другого пользователя

Сайдбар:

- **Инструменты** — только Обнаружение.
- **Администрирование → Настройки** — ADMIN/INFOSEC.
- **Администрирование → Воркфлоу** — все роли, у которых есть пункт (ADMIN тоже, не через Настройки).

Страница «Настройки → Уведомления»

Три отдельные карточки (каждая со своим Save/Discard):

1. **Звуковые уведомления в браузере** — мастер-флаг `browser_sound_enabled`, пресеты по событию (open / resolve / update), опционально свой файл.
2. **Push-уведомления в интерфейсе** — длительность toast для успеха / предупреждения / ошибки (в т.ч. «Пока не закроют»).
3. **Куда настраивать рассылку** — kill-switch `notifications.enabled` и ссылки на Воркфлоу, Интеграции и профили пользователей.

Поля `min_severity` / `notify_on_resolve` / `escalation_*` **не показываются** в UI: runtime их не читает. При миграции политика ушла в системный воркфлоу `legacy_global_policy`.

Deep link: `/settings?section=notifications`.

Роли

Роль	Список воркфлоу	Холст	Kill-switch / Интеграции	Чужие media
ADMIN	Сайдбар → Воркфлоу	да	да	да
INFOSEC	Сайдбар → Воркфлоу (read)	read	read (секреты маскируются)	read (адреса маскируются)
OPERATOR	Сайдбар → Воркфлоу	да	нет	нет (свои в Аккаунте)
VIEWER	Сайдбар → Воркфлоу (доступные)	read	нет	нет; свои media в Аккаунте

Подробнее: Роли.

Системный воркфлоу `legacy_global_policy`

При миграции политика из старых `notifications.*` копируется в системный сценарий с `system_key=legacy_global_policy`. Меняйте опубликованный граф воркфлоу, не глобальные ключи настроек.

Каналы (Интеграции)

Канал	Параметры
Webhook	URL (транспорт notify)
Telegram	bot token, chat id инстанса (fallback)
MAX	bot token, destination (user/chat), id инстанса (fallback)
Email (SMTP)	host, port, SSL/STARTTLS, auth, TLS, trust, timeouts, from/alias/to; пресеты
Jira	URL, email, token, проект — блок **Тикет** создаёт задачу
Yandex Tracker	org, OAuth, очередь — блок **Тикет** создаёт задачу
Другая система (webhook)	имя, URL, секрет — запасной пункт Тикета, не в пикере интеграций

Пробная отправка: **Тест** на карточке интеграции (`POST /api/v1/settings/integrations/{type}/test`) и **Проверить** на канале в профиле (`POST /api/v1/users/me/notification-media/test` или `…/users/{id}/…` для ADMIN). Тест из профиля **не** зависит от kill-switch, чтобы можно было проверить транспорт до включения рассылки.

Звук в браузере (SSE)

Короткий сигнал в открытой вкладке wiSLA. Не заменяет email/Telegram — только локальный feedback оператору.

События (браузерный SSE):

Событие	Когда
`problem.open`	новая проблема
`problem.resolve`	закрытие
`problem.update`	комментарий, severity, назначение и т.п.

Где	Что задаётся
Настройки → Уведомления	Общий вкл/выкл + пресет (или **Выкл**) на каждое событие + один свой файл на инстанс
Аккаунт → Уведомления	Личное: наследовать / вкл / выкл; на каждое событие — наследовать / пресет / выкл / свой файл

Встроенные пресеты (Web Audio): `default`, `alert`, `urgent`, `chime`, `bell`, `horn`, `siren`, `pulse`, плюс `off` и `custom`. Пресеты намеренно различаются по высоте, тембру и ритму. Кнопка **Проверить** играет выбранный в списке черновик (не ждёт Save).

Первый клик по UI после загрузки страницы «разблокирует» AudioContext (ограничение браузера).

Maintenance

На узле в maintenance **новые** problems не открываются, воркфлоу по ним не стартуют.

Карта кода (чтобы переделать / отладить)

Что	Где
Сайдбар: Tools без workflows; Admin + workflows для всех ролей раздела	`frontend/.../layouts/main-layout/main-layout.ts` (`NAV_SECTIONS`)
Страница настроек, секция notifications	`frontend/.../admin/settings-page/settings-page.html` + `.ts`
Список воркфлоу	`frontend/.../workflows/workflow-list-page/`
Редактор холста	`frontend/.../workflows/workflow-editor-page/`
Личные каналы (аккаунт + пользователи)	`frontend/.../account/ui/notification-media-editor/`
Роуты	`frontend/.../workflows/workflows.routes.ts`, `app.routes.ts` (`settings` + `adminGuard`)
API воркфлоу	workflows-api
Faults / events	faults-api
Транспорт / legacy keys	problem-notifications
Настройки инстанса (все секции)	settings.md

Связанные документы

- Воркфлоу (user) — холст, блоки, пресеты
- Воркфлоу (guide) — domain events и правила матчинга
- Frontend: nav и настройки уведомлений — краткий tech-обзор IA

Интеграции

Воркфлоу уведомлений

Воркфлоу уведомлений

Рассылка больше не идёт «всем сразу» из `notifications.min_severity`. Цепочка:

Problem ops → внутренний fault 1:1 → domain event → матчер воркфлоу → блок notify → доставка (SMTP / Telegram / MAX / webhook).

Где открывать UI

Роль	Список сценариев	Холст
Все с доступом	**Администрирование → Воркфлоу** (`/workflows`)	`/workflows/{id}`

ADMIN/INFOSEC также видят ссылку из **Настройки → Уведомления**. Личные адреса — Аккаунт или карточка пользователя.

Операции по проблемам — в События.

Правила матчинга (кратко)

- Envelope: `fault.opened` | `fault.taken` | `fault.reassigned` | `fault.cleared`
- Триггер: `minSeverity=any` (любая критичность), `serviceIds` (пусто = все узлы)
- Блок **Если** после триггера: текущее состояние, выходы Да (`then`) / Нет (`else`). Условия: открыта / не взята / взята / критичность / чем начался запуск / узел. Связка **И/ИЛИ** между условиями (`cases[].join`); без `join` — запасное `combine` блока
- Пауза: минуты + опционально `businessWindows[]` (`day` ISO 1–7, `from`/`to`) или старые `businessDays` / `businessFrom` / `businessTo` (кнопка «По расписанию», timezone инстанса)
- Notify static: `emailAddresses` и `telegramAddresses` (старый `staticAddresses` ещё читается)
- Kill-switch `notifications.enabled=false` глушит **адаптеры**, даже если воркфлоу активны
- При миграции политика `notifications.*` копируется в системный воркфлоу `system_key=legacy_global_policy`

См. Воркфлоу, Уведомления (админ), IA фронта.

Профили каналов

Профили измерений канала

Один канал = один сервис на пути. Один профиль = 1–2 пробы, которые не мешают друг другу.

Scope: профили и каталог ниже — только для `wisla-wiprobe-agent` (софт на хосте). Классический SLAMon/wiProbe ≥ 1.14.10 в списке агентов — инвентарь, не jobs. Зонды в разрыве (WPE / Berkut, SLAMon на железе) — отдельная волна. Матрица: SLAMon ≥ 1.14.10.

Так делают ThousandEyes, Cisco IP SLA, TWAMP/SLAMon: **не вешают все тесты на один объект**. Нагрузка (ёмкость) и непрерывный SLA — разные тесты, часто разные объекты между теми же концами.

Изоляция метрик (hub / star)

На агенте-хабе может быть много channel jobs. Ключи пишутся как:

`channel.{slug}.{c2r|r2c}.s{8hex}.{metric}`

где `s{8hex}` — короткий id сервиса канала. Так last-value и графики не смешивают peer'ов. Backend при чтении отдаёт канонический ключ без токена; планировщики availability/порогов читают **только ключи этого сервиса** (без полного dump `channel.*` по агенту).

Агент: `limits.channel_max_concurrent` по умолчанию **1** — за тик берётся не больше N самых «просроченных» jobs; первый запуск размазан по интервалу (stagger), чтобы хаб не бил ICMP по всем peer разом.

Роль `channel`: с пакета **0.2.17** стоит в дефолтном `agent.yaml` вместе с `host`. Без jobs агент тихий. Старые установки без роли — alpha-channel-enable-roles.py.

Проблемы и доступность канала

Две **разные** проблемы на карточке **канала** (узел/host тут ни при чём — зонд канала может работать без привязки к узлу):

Ключ	Когда	Смысл
`channel.probe.offline`	central и/или remote зонд без heartbeat > **~180 с**	Заголовок: «Один из зондов измерения недоступен»; **какой** зонд — в opdata (`central/remote probe "..."`)
`icmp.unavailable`	**лучший** loss по обеим сторонам ≈ 100% / `probe.error`, **или** метрики старше `wisla.channel.availability.stale-sec` (по умолчанию **900 с**) — **и** все назначенные зонды online	Путь / качество по метрикам

Решение о доступности пути принимает только `ChannelAvailabilityScheduler` (сводка last-value с central **и** remote). Push метрик от одного зонда **не** открывает/закрывает `icmp.unavailable`:

иначе при асимметричном traceroute (одна сторона доходит, другая — `* * *`) лента флапает каждые ~2 мин. Ingest только штампует `last_check_at`.

Backend при чтении last-value / series подставляет ключи профиля канала (не только loss/rtt). Для **TRACE_ROUTE** в KPI должны появляться **Хопы** и **Трасса OK** (не прочерк).

В **списке объектов** бейдж **Trace** (не ICMP): зелёный, если `traceroute.ok=1` хотя бы в одном направлении; красный, если оба направления FAIL; серый — нет свежих метрик. Бейдж **Путь** — **жёлтый (деградация)**, если одно направление c2r/r2c недоступно, а другое OK (раньше был зелёный по «лучшему» направлению). Бейдж **Агент** — **оба** зонда (remote + central): зелёный только когда оба online; offline любого → красный.

Доступность / SLA для канала считается по **лучшему** результату с обеих сторон (min loss): если хотя бы одно направление доходит — канал «доступен», даже когда обратный traceroute с α на ПК даёт `* * *`. Это не ICMP-ping узла, а измерение пути между зондами. Переключатель **«Центр → удалённый»** / **«Удалённый → центр»** показывает метрики **выбранного** направления: на одном может быть OK (18 ms, 2 hop), на другом — FAIL.

Асимметрия c2r / r2c (частая причина «пустой» стороны)

Каждый зонд шлёт пробы **только в свою** сторону: central → `c2r`, remote → `r2c`. Если на карточке одна вкладка пустая, а другая с цифрами — это **не** «нет сбора вообще», а провал **конкретного** направления.

Типичные причины на стенде:

Симптом	Частая причина
ICMP/UDP c2r OK, RFC/TCP/DSCP c2r = 100% loss	На **peer** не открыты inbound-порты echo-блока **8787–8797** (Windows Firewall режет TCP 8787 и UDP 8792/8797, а базовый UDP 8787 — пропускает)
r2c OK, c2r FAIL (или наоборот)	Асимметрия ACL/firewall/NAT между зондами
Обе стороны «Ожидание»	Нет jobs (роль `channel`), зонд offline, или первый цикл ещё не завершился

На агенте echo-сервисы слушают блок портов от `channel.echo_port` (по умолчанию **8787**) + смещения: UDP/TCP **+0**, RFC **+5** (=8792), DSCP **+10** (=8797). Если базовый блок занят, агент выбирает следующий свободный (`8788`, …) и сообщает **`channel_echo_port`** менеджеру; jobs на peer получают порты от **его** base, не от константы 8787.

Проверка на peer: `netstat -an | findstr 8787` (Windows) / `ss -ulnp | grep 8787` (Linux) — после первого channel job должны быть слушатели.

Path MTU на Linux использует raw ICMP с флагом DF (не зависит от flavor `ping`; GNU inetutils без `-M do` раньше давал 100% loss в направлении r2c). Сервис агента по умолчанию — `User=root` в unit; для не-root нужен `CAP_NET_RAW`.

Это не ICMP SLA и **не** Protocol MTU Wisla 5: зонд делает **двоичный поиск** размера с DF (~10 проб за цикл), а не N пакетов фиксированного размера. В параметрах канала задаются **интервал** и **потолок поиска** (ICMP payload или L3 MTU ≥ 1500). Поле «Пакетов» к Path MTU не относится. На карточке KPI — `channel.mtu.bytes` и `channel.mtu.ok`; 11 отправленных — шаги поиска, не «пакеты SLA».

MTU (UDP / Protocol) (`mtu_rfc`) — паритет с Wisla 5 `MTULoadExecutor` / `PROTOCOL:MTU`: UDP load к peer echo, шаги размера, `packet_count` пакетов на шаг, критерий потерь. Метрики: `channel.mtu_rfc.bytes` / `channel.mtu_rfc.ok`. Не путать с Path MTU.

Offline зонда **не** открывает ICMP и **гасит** уже открытый `icmp.unavailable`: пока висит `channel.probe.offline`, loss≈100% / stale обычно значат «нечем мерить», а не подтверждённый обрыв пути.

Почему узлы могут «кричать», а канал молчит дольше по ICMP: при полном сбое агенты перестают пушить пробы, а last-value ещё держит loss=0 — пока не сработает stale. События **«Изменение конфигурации хоста»** относятся к паспорту узла и на карточку канала больше не вешаются.

История статусов (полоса над графиком) красится по порогам шаблона: жёлтый = деградация, красный = отказ. Для RTT берётся худшее из **avg** и **max** (пик max не теряется, если avg ещё «зелёный»). В тултипе графика значения подсвечиваются тем же цветом, что и сегмент истории в этой точке (как на обзоре узла).

Как выбрать

В форме канала, шаг **Измерения** — плитки всех шаблонов + быстрый фильтр; профили сверху подставляют набор. Интервал/пакеты/размер — шаг **Параметры**.

Профиль	Что внутри	Когда
Качество IP	P-Test (ICMP)	Базовый SLA: жива ли линия, потери, задержка. Ставьте почти всегда.
Голос	U-Test (UDP) + MOS	Клиенту важны звонки. Не вместе с ICMP.
Приложения	TCP RTT	Быстро ли поднимается TCP к far-end. HTTP/HTTPS — в доступности услуги.
Маршрут	Traceroute	Разбор инцидента: где обрыв. Не с плотным UDP.
Ёмкость	RFC burst	Приёмка полосы. **Только этот** шаблон на канале.

Каталог слева — рабочие тесты пути: ICMP, UDP, TCP, traceroute, RFC, RFC sustained, DSCP, Path MTU, **MTU UDP (Protocol)**.
DNS / HTTP / HTTPS / SIP — вкладка **Шаблоны → Доступность услуги**.

`TWAMP` (soft subset), `Netflow`, `Y.1731`, `L2`, `OUTER` и похожие hardware / stub типы в операторском каталоге скрыты. Если нужен такой режим, это отдельная волна под железные зонды / специализированные коннекторы, а не обычный канал между `wisla-wiprobe-agent`.

Если нужны и качество, и ёмкость

Создайте **два канала** между теми же зондами:

1. «… качество» → профиль **Качество IP**
2. «… ёмкость» → профиль **Ёмкость** (реже, не круглосуточно плотным burst)

Так метрики не портят друг друга.

Чего не делать

- Все шаблоны каталога на один канал
- RFC вместе с ICMP/UDP на одном канале
- Несколько профилей сразу (клик по карточке заменяет набор)

См. также: Сервисы, Шаблоны каналов.

Веб-сценарий vs availability

Веб-сценарий vs availability vs channel

Кратко:

- **availability** — одна L7/L4 проверка с платформы или одного агента.
- **web_scenario** — journey: ≥1 шаг, общая cookie jar, метрики `web_scenario.*` (время/размер/код как у wiSLA web scenarios, плюс `ssl_cert` ⟶ `web_scenario.ssl.*`).
- **channel** — два зонда, качество пути, метрики `channel.*`.

Шаблон веб-сценария собирается в каталоге (`/templates/web-scenario`: мастер + линейный редактор шагов). Интервал опроса шаблона — **не меньше 60 с** (clamp на сохранении и в планировщике).
Сервис только выбирает шаблон и заполняет URL/креды (или SSL host/port/SNI/пороги). Обзор сервиса: KPI (время, среднее, размер, код) + общий график метрик + цепочка шагов или SSL-панель (карточка UI).

MVP веб-сценария = HTTP/SCRIPT step chain + SSL certificate probe. HTTP-цепочка: платформа **или** связанный агент (≥0.2.27). **BROWSER / headless Chromium** = addon `wisla-browser-runtime` (роль `browser`). ADR-002 (BROWSER (export) collector) не меняется. План режима «Браузер»: Веб-сценарий — Браузер.

Проблемы на `/events`

После platform/probe run:

Ключ	Когда	Severity
`web_scenario.unavailable`	step chain fail (`!ok`) ⟶ open; success ⟶ resolve	`high`
`web_scenario.ssl.*`	недоступность / trust / name / expiry bands	см. SSL thresholds

Имя unavailable включает шаг, если известен (`Web scenario failed (step N)`). Отдельный ключ `web_scenario.step.fail` больше **не открывается** (раньше давал дубликат той же ошибки); при следующем успешном опросе legacy-проблемы с этим ключом закрываются.

Подробнее для операторов: Веб-сценарий.