

Описание процессов жизненного цикла программного обеспечения «INTEBRIX Optimization Platform»

Версия ПО: **1.0.0** (сборка 0ec5bd7d от 21.08.2026) · Правообладатель: ООО «ИНТЕБРИКС» (ИНН 9707059284) · Редакция документа: 19.09.2026

Регламент жизненного цикла платформы для реестра российского ПО: организация разработки, стадии жизненного цикла, версионирование и выпуск релизов, обеспечение качества, устранение неисправностей, обновления, персонал и контакты поддержки — с опорой на фактические средства сборки, тесты и эксплуатационные скрипты репозитория.

1. Общие сведения

Наименование программного обеспечения: INTEBRIX Optimization Platform (Платформа оптимизационного планирования INTEBRIX).

Класс ПО: 08.19 «Программы производственного планирования (APS)» — основной; 09.13 «Средства управления складом и цепочками поставок (WMS, SCM)» — дополнительный. По существу — система поддержки принятия решений на основе математической оптимизации.

Правообладатель: Общество с ограниченной ответственностью «ИНТЕБРИКС».

Реквизит	Значение
Полное наименование	Общество с ограниченной ответственностью «ИНТЕБРИКС»
ИНН	9707059284
КПП	770701001
ОГРН	1267700207895
Дата регистрации	19.06.2026
Адрес	127051, г. Москва, вн.тер.г. муниципальный округ Тверской, ул. Петровка, д. 20/1, помещ. 17
Основной вид деятельности	ОКВЭД 62.01 — Разработка компьютерного программного обеспечения
Адрес электронной почты	info@intebrix.ru
Телефон	+7 916 925-54-71
Сайт	intebrix.ru

Состав программного обеспечения. Платформа включает четыре расчётных ядра — Production Planner (операционное планирование производства, APS), Dispatcher

(маршрутизация и диспетчеризация транспорта, VRP/VRPTW), Strategic Production Planner (объёмное планирование производства, S&OP), Strategic Flow Planner (планирование потоков на логистической сети) — и сквозной контур Supply Chain (Pipeline), связывающий ядра в единую расчётную цепочку.

Технологическая основа. Серверная часть — Python 3.12, FastAPI, Pydantic, SQLAlchemy, Celery; решатели HiGHS и CBC через python-mip; библиотеки NetworkX, NumPy, pandas, SciPy. Клиентская часть — React 19, TypeScript, Vite. Хранилища — PostgreSQL 16, Valkey. Развёртывание — Docker Compose и nginx.

Версия и сборка. К включению в реестр предъявляется версия 1.0.0. Комплект проверочного экземпляра для экспертизы (комплект установки, размещённый для скачивания на сайте правообладателя) собран из сборки, соответствующей фиксации исходного кода 0ec5bd7d от 21.08.2026 (раздел 4.2).

Настоящий документ описывает процессы жизненного цикла программного обеспечения: организацию разработки, стадии жизненного цикла, порядок управления версиями и выпуска обновлений, обеспечение качества, порядок устранения неисправностей, информирование пользователей, требования к персоналу, его численность, фактические адреса размещения и контакты службы поддержки.

2. Организация разработки

2.1 Исполнитель работ

Разработка, развитие, техническая поддержка и сопровождение INTEBRIX Optimization Platform выполняются ООО «ИНТЕБРИКС» силами привлекаемых специалистов — граждан Российской Федерации, работающих по гражданско-правовым договорам (внештатные специалисты); работников по трудовым договорам у правообладателя нет. Работы по проектированию, программированию, тестированию, выпуску версий и сопровождению сторонним организациям не передаются. Численность и роли специалистов приведены в разделе 8.3, фактические адреса размещения — в разделе 8.4.

Исключительное право на программу для ЭВМ принадлежит ООО «ИНТЕБРИКС» в полном объёме: право приобретено у автора по договору об отчуждении исключительного права № 1 от 22.06.2026; программа зарегистрирована в Роспатенте (свидетельство о государственной регистрации программы для ЭВМ № 2026684186 от 09.08.2026). Объект «INTEBRIX Optimization Platform» (компьютерная программа, дата создания 10.05.2025) депонирован в Распределённом реестре интеллектуальных прав РЦИС.РФ.

Все работы ведутся на территории Российской Федерации.

2.2 Хранение исходного кода

Исходный код платформы ведётся в системе контроля версий Git в едином репозитории, включающем серверную часть (backend/), клиентские приложения (apps/, packages/), инфраструктуру развёртывания (docker/, docker-compose.yml), эксплуатационные скрипты (scripts/) и документацию (docs/). История изменений ведётся с 08.03.2026 и

содержит полную последовательность правок с указанием автора и даты; с этой даты исходный код ведётся в текущей структуре единого репозитория. Разработка программы начата автором 10.05.2025 (раздел 2.1); исключительное право на программу, включая результаты ранних этапов разработки, передано правообладателю по договору об отчуждении № 1 от 22.06.2026 с актом приёма-передачи. Доработки программы, выполняемые автором после передачи исключительного права, принадлежат правообладателю по условиям договора с автором.

Технические средства хранения исходного кода находятся на территории Российской Федерации:

Средство хранения	Место нахождения
Рабочая копия репозитория — рабочая станция правообладателя (собственное оборудование)	127051, г. Москва, ул. Петровка, д. 20/1, помещ. 17
Удалённая копия репозитория — российский сервис хостинга исходного кода GitFlic (gitflic.ru), проект info-intebrix-ru/intebrix, доступ закрытый	оператор сервиса — ООО «РеСолют» (ИНН 9704054697, ОГРН 1217700127094, г. Москва, Огородный проезд, д. 16/1, стр. 5); сервис включён в единый реестр российских программ для ЭВМ и баз данных (реестровая запись № 15861 от 09.12.2022); серверные мощности сервиса расположены на территории Российской Федерации

Удалённая копия ведётся на сервисе GitFlic с 19.08.2026. Выпущенной версии соответствует аннотированная метка v1.0.0; идентичность рабочей и удалённой копий подтверждается сверкой идентификаторов фиксаций. Подробное описание технических средств хранения приведено в письме о технических средствах хранения и компиляции (документ 07 комплекта).

2.3 Средства сборки

Сборка (компиляция исходного текста в объектный код — образы контейнеров) выполняется на рабочей станции правообладателя по адресу, указанному в разделе 2.2, двумя взаимодополняющими способами:

1. **Сборка образов контейнеров** — `docker compose build` по файлам `docker/Dockerfile.api`, `docker/Dockerfile.worker`, `docker/Dockerfile.frontend`; базовые образы `python:3.12-slim`, `node:20-alpine`, `nginx:1.27-alpine`, `postgres:16-alpine`, `valkey/valkey:8-alpine`. Целевая платформа образов — `linux/amd64`; при сборке на рабочей станции другой архитектуры используется `docker buildx` с явным указанием платформы.
2. **Сборка дистрибутива для поставки** — скрипт `scripts/build-airgap.sh`, который выполняет сборку образов под целевую платформу и формирует единый

архив `intebrix-airgap-<версия>.tar.gz`, включающий сохранённые образы контейнеров (`docker save`), файл `docker-compose.yml`, файлы сборки образов и конфигурацию `nginx`, шаблон конфигурации `.env.example`, эксплуатационные скрипты, перечень сторонних компонентов `THIRD-PARTY-NOTICES.txt`, файл `VERSION`, памятку `README.md` и контрольные суммы `SHA256SUMS` для контроля целостности. Рядом с архивом размещается распаковщик `intebrix-install.sh`.

Все проверки раздела 5, по результатам которых принимается решение о выпуске версии, выполняются на рабочей станции правообладателя до отправки изменений в удалённую копию репозитория и перед выпуском версии; распространяемые образы и комплект поставки собираются на той же рабочей станции.

Работоспособность процедур установки на российских операционных системах подтверждена испытаниями 18.08.2026 (документ 05 «Протокол испытаний на российских операционных системах»). Испытания проводились на временных арендованных виртуальных серверах на территории Российской Федерации; на стенде Astra Linux Special Edition 1.8.2 образы собирались на самом стенде для проверки пути установки заказчика в подключённом контуре, на стенде РЕД ОС 8.0.2 выполнялось развёртывание из комплекта закрытого контура. Испытательные стенды средством сборки распространяемого дистрибутива не являются.

2.4 Компонентный состав и лицензионная чистота

Состав используемых сторонних компонентов с открытым исходным кодом и их лицензии зафиксированы в файле `THIRD-PARTY-NOTICES.txt`, формируемом скриптом `scripts/gen_third_party_notices.py`, и в документе `docs/bill-of-materials.md`. При выборе компонентов исключается использование лицензий копилефт-типа, распространяющих требования на производный продукт: в частности, подсистема кэширования переведена на `valkey/valkey:8-alpine` (лицензия BSD-3) с фиксацией запрета на возврат к образам под лицензиями `RSALv2/SSPL/AGPLv3`. Порядок приобретения сторонних компонентов описан в разделе 3.8.

3. Стадии жизненного цикла

Жизненный цикл платформы организован по итеративной модели: развитие ведётся последовательными этапами (фазами и спринтами), каждый из которых завершается выпуском версии. Соответствие процессов жизненного цикла разделам настоящего документа:

Процесс	Разделы документа
Проектирование	3.1, 3.2
Разработка	3.3, 4
Тестирование	3.4, 5
Приобретение	3.8
Поставка (передача заказчику)	3.9, 7.1

Процесс	Разделы документа
Эксплуатация	3.6
Документирование	3.10
Обучение и квалификация персонала	3.11, 8
Поддержка версий и доработка	4, 7
Устранение сбойных ситуаций	6
Снятие с сопровождения и вывод из эксплуатации	3.7

Ниже перечислены стадии и работы, выполняемые на каждой из них.

3.1 Сбор и анализ требований

Источники требований:

- задачи и замечания заказчиков, поступающие при внедрении и в ходе эксплуатации;
- результаты внутренних аудитов продукта (функционального, технического, аудита безопасности, аудита производительности), оформляемые в каталоге `audit/` и в документах `docs/`;
- требования нормативных актов Российской Федерации, применимые к обработке данных и к отечественному программному обеспечению;
- собственный план развития продукта.

Результат стадии: зафиксированный перечень требований с приоритетами. Требования ведутся в документах плана развития (`docs/backlog-roadmap.md`, `docs/post-phase-h-roadmap.md`, планы работ по отдельным направлениям — например, `docs/plan-performance-2026-08.md`, `docs/plan-security-2026-08.md`, `docs/plan-licensing-legal-2026-08.md`). Каждому требованию присваивается идентификатор вида `P0-8 #4`, где префикс обозначает приоритет (`P0` — блокирующий, `P1` — высокий, `P2` — плановый), и этот идентификатор сопровождает требование до записи в журнале изменений.

3.2 Проектирование

Работы: уточнение архитектуры затрагиваемой подсистемы, определение изменений схемы данных, определение изменений программного интерфейса (API), оценка влияния на смежные подсистемы и на совместимость с уже развёрнутыми установками.

Результат стадии: проектные решения, зафиксированные в документации архитектуры (`docs/architecture.md`, `docs/platform-technical-overview.md`), описание программного интерфейса (`docs/api-reference.md`), соглашения по пользовательскому интерфейсу (`docs/ui-conventions.md`), при изменении схемы данных — план миграции.

3.3 Разработка

Работы: внесение изменений в исходный код в отдельной ветке системы контроля версий; изменения схемы базы данных оформляются миграциями Alembic (`backend/alembic/versions/`); изменения программного интерфейса сопровождаются моделями Pydantic, обеспечивающими валидацию входных и выходных данных.

Правила ведения кода:

- изменения выполняются в рабочих ветках, интеграция в основную ветку `main` — через запрос на слияние;
- каждое изменение сопровождается сообщением фиксации с указанием сути правки;
- на каждое изменение, устраняющее дефект, создаётся автоматический тест, воспроизводящий дефект (регрессионный тест);
- статический анализ и проверки раздела 5.3 выполняются на рабочей станции до отправки изменений в удалённую копию репозитория.

3.4 Тестирование

Состав и порядок проверок описаны в разделе 5 настоящего документа.

3.5 Выпуск версии

Работы: прогон полного набора проверок; сборка образов контейнеров; при поставке в закрытый контур — формирование архива дистрибутива скриптом `scripts/build-airgap.sh` с контрольными суммами; внесение записи в журнал изменений `docs/CHANGELOG.md`; актуализация эксплуатационной документации (раздел 3.10).

Критерии готовности к выпуску приведены в разделе 5.5.

3.6 Сопровождение (эксплуатация)

Работы: приём и обработка обращений пользователей (раздел 6); выпуск исправляющих версий; консультации по эксплуатации; помощь при установке и обновлении; наблюдение за состоянием установок посредством штатных средств контроля.

Штатные средства контроля состояния установки:

Средство	Назначение
GET /health/live	проверка работоспособности процесса API
GET /health/ready	проверка готовности к обслуживанию (доступность БД, кэша, очереди)
GET /metrics	метрики в формате Prometheus, в том числе счётчик расчётов <code>solver_runs_total{core,status}</code>

Средство	Назначение
<code>POST /api/client-errors</code>	приём необработанных ошибок клиентской части с записью в <code>logs/client_errors.jsonl</code> (работает в закрытом контуре без обращения к внешним сервисам)
<code>logs/audit.jsonl</code>	журнал действий пользователей (ротация 50 МБ × 5 файлов)
<code>logs/app.jsonl</code>	журнал жизненного цикла приложения (ротация 20 МБ × 5 файлов)
<code>docker-compose.glitchtip.yml</code>	опциональная развёртываемая в контуре заказчика система сбора ошибок

Резервное копирование и восстановление входят в состав поставки: `scripts/backup.sh` формирует единый архив (логическая выгрузка PostgreSQL, журналы аудита и клиентских ошибок за 30 дней, конфигурация, манифест с версией и меткой времени; при указании переменной `INTEBRIX_BACKUP_GPG_RECIPIENT` архив шифруется GPG), `scripts/restore.sh` выполняет обратную операцию.

3.7 Снятие с сопровождения и вывод из эксплуатации

Снятие версии продукта с сопровождения. Предыдущая мажорная версия поддерживается не менее 12 месяцев после выпуска новой мажорной версии. О прекращении поддержки версии пользователи уведомляются не позднее чем за 90 календарных дней по адресу электронной почты, указанному в договоре. По запросу заказчика предоставляются финальная версия дистрибутива снимаемой с поддержки версии и соответствующая ей документация. В течение срока поддержки прежней мажорной версии для неё выпускаются исправления безопасности и критических дефектов; новая функциональность выпускается только в текущей мажорной версии.

Прекращение использования конкретной установки. В составе платформы реализованы функции выгрузки и удаления данных организации-пользователя:

- `POST /api/admin/export-org` — формирование выгрузки данных организации в виде архива JSONL-файлов (организации, пользователи, модели, расчёты, журналы аудита, справочники Data Hub, ключи активации, настройки коннекторов, webhook-подписки, ключи доступа к API; секреты в выгрузке маскируются). Операция выполняется в фоне, состояние и ссылка на результат доступны через `GET /api/admin/exports/{export_id}`;
- `DELETE /api/admin/org/{id}` — постановка организации в очередь на удаление с отложенным исполнением через 24 часа; при включённой многофакторной аутентификации требуется подтверждение кодом TOTP; перед каскадным удалением журнал аудита архивируется в `logs/deleted_orgs/{org_id}-{ts}.jsonl`;

- `DELETE /api/admin/org/{id}/cancel-pending` — отмена запланированного удаления в течение отсрочки.

Для данных расчётов действует настраиваемое ограничение срока хранения (автоматическое усечение таблицы расчётов до заданного предела, `services/solver_runner`). Сроки хранения журналов аудита определяются условиями договора.

3.8 Приобретение

Правообладатель приобретает (получает) для создания и поставки платформы следующие виды ресурсов:

- **сторонние программные компоненты** — библиотеки Python и Node.js, базовые образы контейнеров, решатели HiGHS (лицензия MIT) и CBC (лицензия EPL-2.0). Компоненты используются только на условиях открытых лицензий, не налагающих ограничений на распространение и использование на территории Российской Федерации и не распространяющих требования на собственный код продукта; проприетарные компоненты в состав поставки не приобретаются и не входят. Порядок включения нового компонента: обоснование потребности, проверка лицензии на совместимость с условиями поставки, закрепление версии в файлах `backend/requirements.lock` и `package-lock.json`, внесение в `THIRD-PARTY-NOTICES.txt` (раздел 2.4). Компоненты получают на рабочую станцию правообладателя при сборке; в комплект закрытого контура включаются готовые образы, поэтому заказчику доступ к внешним репозиториям не требуется;
- **услуги** — хостинг удалённой копии репозитория (сервис GitFlic, раздел 2.2) и вычислительные мощности российского облачного провайдера (Yandex Cloud, регион `ru-central1`, Российская Федерация) для предоставления доступа к платформе как к интернет-сервису и для размещения сайта правообладателя `intebrix.ru`, на котором опубликован комплект установки проверочного экземпляра. Услуги приобретаются у российских поставщиков с размещением серверов на территории Российской Федерации;
- **оборудование** — рабочая станция сборки и хранения кода является собственным оборудованием правообладателя.

3.9 Поставка (передача заказчику)

Поставка выполняется одним из способов, предусмотренных тарифной политикой правообладателя:

1. **Размещение в инфраструктуре заказчика** (основной способ) — передача комплекта закрытого контура `intebrix-airgap-<версия>.tar.gz` с распаковщиком `intebrix-install.sh` согласованным с заказчиком каналом (защищённая ссылка правообладателя либо носитель информации). Перечень контрольных сумм файлов входит в комплект (SHA256SUMS), контрольная сумма самого архива сообщается заказчику отдельным каналом связи; заказчик проверяет целостность перед установкой. Установка выполняется по инструкции по

установке (документ 02 комплекта). При размещении в подключённом контуре допускается установка из исходного дистрибутива по той же инструкции.

2. **Предоставление доступа к платформе через сеть «Интернет»** (дополнительный способ) — на вычислительных мощностях правообладателя, расположенных на территории Российской Федерации; заказчику передаются учётные записи администратора организации.

Вместе с дистрибутивом заказчику передаются эксплуатационная документация (раздел 3.10) и лицензионный документ (порядок выпуска и передачи описан в письме о технических средствах — документ 07 комплекта, раздел 2). Передача обновлений описана в разделе 7.1.

3.10 Документирование

Состав документации на версию: описание функциональных характеристик, инструкция по установке, руководство по эксплуатации (руководство пользователя и администратора), настоящее описание процессов жизненного цикла, описание технической архитектуры. Внутренняя техническая документация ведётся в каталоге docs/ репозитория (раздел 10); описание программного интерфейса в формате OpenAPI формируется серверной частью автоматически из моделей Pydantic.

Порядок ведения. Документация ведётся на русском языке в исходном формате Markdown в том же репозитории, что и код, и изменяется в той же рабочей ветке, что и изменение функциональности; актуализация эксплуатационной документации является обязательным шагом выпуска версии (раздел 4.3, шаг 5). Каждый документ содержит номер версии ПО, к которой относится, и дату редакции. Ответственность за актуальность документа несёт специалист, вносящий изменение; проверка полноты документации выполняется при подготовке выпуска.

Публикация. Комплект эксплуатационной документации на текущую версию размещается на сайте правообладателя по адресу <https://intebrix.ru/docs>; журнал выпусков — по адресу <https://intebrix.ru/changelog>.

3.11 Обучение и квалификация персонала

Персонал правообладателя. Требования к квалификации приведены в разделе 8.2 и проверяются при привлечении специалиста. Вводное ознакомление включает изучение документации архитектуры и эксплуатационной документации, правил ведения кода (раздел 3.3) и состава проверок (раздел 5); изменения нового специалиста проходят рассмотрение через запросы на слияние. При выпуске мажорной версии и при изменении процедур установки или сопровождения специалисты сопровождения знакомятся с изменениями до начала поставки версии заказчикам.

Персонал заказчика. Базовое обучение администраторов и пользователей заказчика входит в состав работ по внедрению и проводится на основе руководства по эксплуатации; обучение сверх базового выполняется по отдельному соглашению (тарифная политика правообладателя, п. 4.7).

4. Управление версиями и релизами

4.1 Система контроля версий

Исходный код ведётся в Git. Основная ветка — `main`; работы выполняются в отдельных ветках и переносятся в основную ветку через запрос на слияние с прохождением проверок.

4.2 Нумерация версий и выпуски

Применяется трёхкомпонентная нумерация вида **МАЖОРНАЯ.МИНОРНАЯ.ПАТЧ** (X.Y.Z):

Компонент	Когда увеличивается
МАЖОРНАЯ (X)	несовместимые изменения программного интерфейса или формата данных, требующие изменений на стороне интегрированных систем заказчика
МИНОРНАЯ (Y)	новая функциональность с сохранением обратной совместимости
ПАТЧ (Z)	устранение дефектов, исправления безопасности, изменения, не затрагивающие функциональный состав

Номер версии задаётся единственным файлом `VERSION` в корне репозитория; из него значение получают конфигурация серверной части (`app_version`, возвращается в ответах `/health` и в описании OpenAPI), имена образов контейнеров, наименование дистрибутива закрытого контура и манифесты клиентской части. Совпадение номера во всех артефактах поставки проверяется автоматически перед выпуском (`scripts/release-check.sh`, проверка 2 «Единый номер версии»).

Выпущенные версии поставки. Первая коммерческая версия — **1.0.0**: единая нумерация версий поставки, лицензирование на подписанных документах, испытания совместимости с российскими операционными системами, комплект закрытого контура. Под номером 1.0.0 выпущены две сборки:

Сборка	Дата	Обозначение в системе контроля версий	Применение
Первая	19.08.2026	аннотированная метка <code>v1.0.0</code> (фиксация <code>38d1acae</code>)	сборка, на которой 18.08.2026 проведены испытания на

Сборка	Дата	Обозначение в системе контроля версий	Применение
			русских операционных системах (документ 05 комплекта)
Вторая	21.08.2026	фиксация 0ec5bd7d	комплект проверочного экземпляра для экспертизы (комплект установки, собран 19.09.2026)

Во второй сборке по сравнению с первой изменены: расчётное ядро стратегического планирования производства (`backend/app/solvers/generic_layer_solver.py` — исправления обработки правил конструктора ограничений, моделей без слоя оборудования и направления цели), компонент конструктора ограничений клиентской части

(`apps/shell/src/cores/strategic-production/components/ConstraintBuilder.jsx`), добавлены три файла регрессионных тестов (`test_c3_constraint_dialect.py`, `test_c3_continuous_branch.py`, `test_c3_schema_null_defaults.py`) и дополнен журнал изменений. Состав поставки, сторонние зависимости, процедура установки и номер версии не изменялись. Начиная со следующего выпуска каждая сборка, передаваемая заказчиком, получает новый номер версии.

Внутренние этапы разработки до первого выпуска. В журнале изменений `docs/CHANGELOG.md` записи с номерами 3.0.0 (21.03.2026), 3.0.1–3.0.5 (06.05.2026) и 3.1.0 (07.05.2026) относятся к внутренней нумерации этапов разработки, предшествовавшей первому коммерческому выпуску; версиями поставки они не являются и заказчикам не передавались.

Журнал изменений ведётся в формате Кеер а Changelog 1.1.0: для каждой версии перечисляются добавленное, изменённое и исправленное с указанием идентификаторов исходных требований.

Обозначение выпуска в системе управления версиями. На каждый выпуск проставляется аннотированная метка вида `vX.Y.Z`, совпадающая со значением файла `VERSION`. Метки этого вида обозначают выпущенные версии и не используются для обозначения контрольных точек работ.

4.3 Порядок выпуска версии

1. Завершение работ по составу версии и прохождение проверок раздела 5.

2. Внесение записи в `docs/CHANGELOG.md` с перечнем изменений, несовместимых изменений и известных ограничений.
3. Обновление номера версии в файле `VERSION` и проверка согласованности `scripts/release-check.sh`.
4. Сборка образов контейнеров; для поставки в закрытый контур — формирование архива дистрибутива с контрольными суммами SHA-256.
5. Актуализация эксплуатационной документации (описание функциональных характеристик, инструкция по установке, руководство по эксплуатации, описание программного интерфейса).
6. Публикация уведомления о выпуске (раздел 7).

4.4 Изменения схемы базы данных

Все изменения схемы оформляются миграциями Alembic и применяются командой `alembic upgrade head`. Текущая цепочка миграций: `001_baseline` (базовая схема), `002_audit_logs_anonymize` (обезличивание записей журнала аудита), `003_dispatch_days` (сменные задания диспетчеризации), `004_license_grant` (хранение подписанного лицензионного документа как источника сведений о лицензии). Применение миграций встроено в процедуру обновления (раздел 7.1) и выполняется после создания резервной копии.

5. Обеспечение качества

5.1 Автоматизированные проверки серверной части

Набор автоматических тестов серверной части расположен в каталоге `backend/tests` и выполняется средством `pytest`. Состав в сборке `0ec5bd7d` от 21.08.2026 — **62 файла тестов, 534 тестовые функции**; по сравнению с первой сборкой версии 1.0.0, для которой протокол испытаний (документ 05) зафиксировал 608 проверок с учётом параметризации, добавлены три файла тестов расчётного ядра стратегического планирования производства (на 36 проверок больше).

Виды тестирования и соответствующие наборы:

Вид тестирования	Наборы тестов
Модульное тестирование расчётной логики	<code>test_solvers.py</code> , <code>test_cost_model.py</code> , <code>test_conflict_detector.py</code> , <code>test_fleet_availability.py</code> , <code>test_rolling_availability.py</code> , <code>test_dispatch_day.py</code>
Регрессионное тестирование расчётных ядер	<code>test_c2_engine_regression.py</code> (набор регрессионных сценариев планировщика производства), <code>test_dispatcher_phase_h.py</code> , <code>test_dispatcher_backhaul.py</code> ,

Вид тестирования	Наборы тестов
	<code>test_c3_constraint_dialect.py</code> , <code>test_c3_continuous_branch.py</code> , <code>test_c3_schema_null_defaults.py</code> (регрессионные тесты ядра стратегического планирования производства, добавлены в сборке от 21.08.2026)
Проверка корректности решений	<code>test_dispatcher_vrptw_correctness.py</code> — проверка соблюдения временных окон, вместимости и связности маршрутов
Проверка детерминированности	<code>test_solver_determinism.py</code> — на намеренно симметричной модели с множеством равнооптимальных решений проверяется, что один и тот же вход даёт идентичный результат между запусками
Проверка воспроизводимости на данных заказчика	наборы проверки воспроизводимости расчётов на обезличенных производственных данных отраслевого кейса
Эталонное (golden) тестирование результатов	<code>test_perf_golden.py</code> с эталоном <code>tests/fixtures/perf_golden.json</code> — для каждого расчётного ядра вычисляется устойчивый отпечаток значимых полей ответа и сверяется с эталоном; набор обязателен к прогону при любых работах по оптимизации быстродействия и не допускает молчаливого изменения результата расчёта
Тестирование программного интерфейса (сквозное, по HTTP)	<code>test_api.py</code> , <code>test_dispatcher_http_e2e.py</code> , <code>test_input_echo_e2e.py</code> , <code>test_tenant_lifecycle_e2e.py</code> , <code>test_erase_pii_e2e.py</code>
Тестирование изоляции данных между организациями	<code>test_cross_tenant_isolation.py</code> , <code>test_tenant_filter_linter.py</code> , <code>test_tenant_settings.py</code>
Тестирование разграничения	<code>test_api_keys_scopes.py</code> ,

Вид тестирования	Наборы тестов
доступа	<code>test_seed_guard.py</code>
Тестирование лицензирования	<code>test_license_coverage.py</code> , <code>test_license_delivery.py</code> , <code>test_license_editions.py</code> , <code>test_license_enforcement.py</code> , <code>test_license_import_robustness.py</code> , <code>test_grant.py</code>
Тестирование валидации входных данных	<code>test_datahub_validation.py</code> , <code>test_datahub_preview.py</code> , <code>test_empty_input_cross_core.py</code> , <code>test_dispatcher_sync_size_guard.py</code>
Тестирование обработки ошибок клиента	<code>test_client_errors.py</code>
Тестирование сроков хранения данных	<code>test_solver_runs_retention.py</code>
Тестирование шаблонов уведомлений	<code>test_email_templates.py</code>

5.2 Автоматизированные проверки пользовательского интерфейса

Сквозные проверки интерфейса выполняются средством Playwright (`playwright.config.ts`, каталог `e2e/`). Состав в сборке от 21.08.2026 — **18 файлов сценариев, 76 сценариев** (по перечню Playwright для одного браузерного профиля):

- навигация и маршрутизация приложения (`navigation.spec.ts`);
- работа расчётных ядер через интерфейс (`cores.spec.ts`, `solver-flow.spec.ts`, `m2v1-c1-from-blank.spec.ts`, `m2v2-c1-form-suggest.spec.ts`, `m68-c4-multimodal.spec.ts`);
- загрузка и разбор данных (`datahub.spec.ts`, `datahub-flow.spec.ts`);
- работа без доступа к внешним сервисам (`m2v3-fully-offline.spec.ts`) — проверка применимости в закрытом контуре;
- безопасность клиентской части: хранение сессии (`auth-cookie.spec.ts`), передача авторизации на экранах приложения (`auth-header-screens.spec.ts`), заголовки безопасности HTTP (`security-headers.spec.ts`), экранирование данных в отчёте для руководителя (`exec-report-xss.spec.ts`);
- доступность интерфейса (`accessibility.spec.ts` с использованием `@axe-core/playwright`), управление с клавиатуры (`keyboard.spec.ts`);
- оформление и визуальные регрессии (`theme.spec.ts`, `visual.spec.ts`), эталонный регресс картографического слоя (`maps-golden.spec.ts`).

Матрица браузеров при полном прогоне: Chromium (покрывает Chrome и Edge), Firefox, WebKit (Safari), мобильный профиль iPhone 13. Локальный прогон по умолчанию

выполняется на Chromium, полная матрица включается переменной окружения. При падении сценария сохраняется снимок экрана.

5.3 Статические проверки и проверки безопасности

Проверки выполняются на рабочей станции правообладателя при каждом изменении исходного кода (до отправки в удалённую копию репозитория) и повторяются перед выпуском версии:

- **Статический анализ серверной части** — анализатор `ruff` фиксированной версии; конфигурация набора правил хранится в репозитории (`backend/ruff.toml`), версия анализатора закреплена в `backend/requirements-dev.txt` (`ruff==0.12.0`). Отрицательный результат останавливает выпуск.
- **Проверка типов клиентской части** — средствами компилятора TypeScript (`scripts/typecheck.sh`). Проверка выполнена в виде ограничителя: зафиксировано текущее число замечаний, и проверка считается отрицательной при его превышении. Такой порядок принят потому, что значительная часть замечаний относится к настройке компилятора и к расхождению с типами сторонних библиотек после обновления их версий; требование немедленного нулевого результата привело бы к постоянно отрицательной проверке, которую перестают читать. Порог снижается вместе с устранением замечаний.
- **Проверка разграничения данных между организациями** — `backend/tools/check_tenant_filters.py`: анализ синтаксического дерева всех обработчиков программного интерфейса; выпуск останавливается при обнаружении выборки из таблицы, привязанной к организации, без фильтра по `organization_id`. Проверка исключает появление обработчика, раскрывающего данные чужой организации.
- **Проверка целостности сборки** — контрольный импорт приложения с подсчётом зарегистрированных маршрутов; проверка работоспособности модулей аутентификации (выпуск и разбор токена доступа, хеширование и проверка пароля).
- **Проверка доступности и качества клиентской части** — Lighthouse CI (`lighthouseci.json`) на публичных маршрутах приложения, по два прогона на маршрут. Пороговые значения: доступность не ниже 0,90 — блокирующее условие; контрастность текста, наличие альтернативного текста изображений, наличие подписей полей форм, наличие атрибута языка документа — блокирующие условия. Быстродействие (не ниже 0,85), соответствие лучшим практикам (не ниже 0,90) и поисковая оптимизация (не ниже 0,85) проверяются как предупреждения.
- **Контроль состава сторонних компонентов и лицензий** — формирование `THIRD-PARTY-NOTICES.txt` скриптом `scripts/gen_third_party_notices.py`, проверка отсутствия компонентов под лицензиями, несовместимыми с условиями поставки.
- **Проверка готовности к выпуску** — `scripts/release-check.sh`: отсутствие ключа подписи лицензий разработки в поставке, единый номер версии, работоспособность примера окружения, независимость имён образов от каталога

сборки, отсутствие секретов и небезопасных умолчаний в поставке, наличие уведомлений о сторонних компонентах.

5.4 Нагрузочное тестирование

Нагрузочные сценарии выполняются средством k6 (`load_tests/c1_dispatcher.js`), запускаемым в контейнере без установки на рабочую станцию. Профиль нагрузки и длительность задаются переменными окружения. Результат последнего зафиксированного прогона (внутренний этап 3.0.4): 100% успешных проверок, 95-й процентиль времени расчёта — 0,67 с, 95-й процентиль времени сквозного запроса — 137 мс.

5.5 Проверки перед выпуском версии

Перед выпуском версии на рабочей станции правообладателя выполняются:

1. Сборка клиентской части (установка зависимостей `npm ci`, сборка оболочки приложения).
2. Сборка и контрольный импорт серверной части на Python 3.12; проверка модулей аутентификации.
3. Прогон автоматических тестов серверной части, не требующих развёрнутой базы данных (наборы: контроль первичного наполнения, обработка пустого ввода всеми ядрами, области действия ключей API, настройки арендатора, разбор и валидация загружаемых данных, проверка фильтров арендатора, шаблоны уведомлений, ограничение размера синхронного запроса к диспетчеризации).
4. Статические проверки раздела 5.3, включая проверку фильтров разграничения данных между организациями и проверку готовности к выпуску.
5. Прогон эталонного (golden) набора `test_perf_golden.py` — обязателен при любых изменениях, затрагивающих расчётную логику или быстродействие.
6. Прогон сквозных сценариев интерфейса на полной матрице браузеров.
7. Проверка доступности клиентской части (Lighthouse CI).
8. Сборка трёх образов контейнеров (API, фоновый обработчик, клиентская часть) под целевую платформу `linux/amd64`.
9. Контрольная установка собранного дистрибутива на чистом окружении и проверка отклика `/health/live` и `/health/ready`.

Выпуск версии не производится при отрицательном результате любой из проверок 1–8.

Полный прогон набора тестов на развёрнутом стенде. Перед выпуском выполняется полный набор автоматических проверок серверной части, включая проверки, требующие развёрнутой базы данных и очереди задач. Прогон выполняется на стенде, развёрнутом по документированной процедуре, а не в окружении разработчика. Результат прогона на целевых операционных системах фиксируется протоколом (для версии 1.0.0 — документ 05 «Протокол испытаний на российских операционных системах» от 18.08.2026).

6. Устранение неисправностей

6.1 Приём обращений

Обращения принимаются:

- по электронной почте support@intebrix.ru — основной канал приёма обращений (общие вопросы — info@intebrix.ru);
- по телефону +7 916 925-54-71 в режиме работы службы поддержки (раздел 9);
- через форму обратной связи на сайте intebrix.ru (раздел «Контакты»);
- по выделенному каналу связи, создаваемому для заказчика на этапе внедрения (при наличии соответствующего условия в договоре).

Обращение регистрируется с фиксацией даты и времени поступления, наименования организации-заявителя, контактного лица, версии продукта, режима развёртывания и описания проблемы.

Состав сведений, прилагаемых к обращению (запрашивается у заявителя для ускорения разбора):

- номер версии платформы (возвращается запросом GET /health либо отображается в интерфейсе);
- режим развёртывания (cloud — размещение на мощностях правообладателя, on_prem — инфраструктура заказчика, air_gapped — изолированный контур);
- последовательность действий, приводящая к проблеме;
- фрагменты журналов logs/app.jsonl, logs/audit.jsonl, logs/client_errors.jsonl за период возникновения проблемы, либо вывод команд docker compose logs api и docker compose logs worker;
- при проблемах с расчётом — исходные данные модели или их обезличенный фрагмент.

6.2 Классификация обращений по критичности

Уровень	Признак	Примеры
Критический	платформа недоступна либо основная функция полностью неработоспособна; отсутствует обходной путь	сервис не отвечает; расчёт не запускается ни на одном ядре; невозможен вход пользователей; выявлено нарушение разграничения доступа к данным
Высокий	существенное ухудшение работы; функция работоспособна частично либо	одно из расчётных ядер завершается ошибкой на типовых данных; отказ импорта данных; значительная

Уровень	Признак	Примеры
Обычный	доступен трудоёмкий обходной путь	деградация времени расчёта
	ограниченное неудобство, не препятствующее работе; вопрос по эксплуатации; запрос на доработку	некорректное отображение элемента интерфейса; вопрос по настройке ограничений модели; предложение по развитию

6.3 Сроки реакции и устранения

Значения утверждены правообладателем 19.08.2026. Обращения принимаются в рабочие дни с 10:00 до 19:00 по московскому времени; сроки исчисляются в пределах этого режима. Обращение, поступившее вне режима работы, регистрируется, отсчёт срока реакции начинается с ближайшего рабочего часа. Повышенные уровни обслуживания (включая расширенный режим приёма обращений) предоставляются по отдельному соглашению о сопровождении.

Уровень	Срок реакции	Срок устранения
Критический	не более 4 рабочих часов	восстановление работоспособности (в том числе обходным решением) — не более 1 рабочего дня; полное устранение — во внеочередном исправляющем выпуске
Высокий	не более 8 рабочих часов	не более 5 рабочих дней либо в ближайшем плановом выпуске
Обычный	не более 2 рабочих дней	в плановом порядке, с сообщением заявителю о принятом решении

Показатель доступности при предоставлении доступа к платформе как к интернет-сервису на мощностях правообладателя: не менее **99,5%** в календарный месяц, без учёта заранее согласованных окон обслуживания. Повышенный показатель может устанавливаться отдельным соглашением. Для установок в инфраструктуре заказчика показатель доступности определяется заказчиком и условиями договора сопровождения.

6.4 Порядок обработки

1. **Регистрация.** Обращение фиксируется, заявителю направляется подтверждение приёма.

2. **Классификация.** Определяется уровень критичности по критериям раздела 6.2.
3. **Воспроизведение.** Дефект воспроизводится на стенде разработчика на данных, предоставленных заявителем, либо на эквивалентном наборе.
4. **Локализация.** Определяется причина дефекта в исходном коде или конфигурации.
5. **Разработка исправления и регрессионного теста.** На каждый подтверждённый дефект создаётся автоматический тест, воспроизводящий дефект до исправления и проходящий после него. Тест включается в постоянный набор (`backend/tests`), что исключает повторное появление дефекта в последующих версиях.
6. **Проверка.** Выполняется набор проверок раздела 5, включая эталонный (`golden`) набор.
7. **Выпуск.** Исправление включается в очередную версию: критические дефекты — во внеочередную патч-версию, остальные — в плановую.
8. **Уведомление заявителя.** Заявителю сообщается номер версии, содержащей исправление, и порядок её применения.
9. **Обходное решение.** До выпуска исправления, при наличии технической возможности, заявителю предоставляется описание временного обходного пути.

6.5 Обращения, не относящиеся к дефектам

Обращения, связанные с ошибками в исходных данных заказчика, с настройкой модели планирования, с интеграцией со сторонними системами или с запросом новой функциональности, обрабатываются как консультации либо как заявки на доработку. Заявки на доработку включаются в план развития продукта; при существенном объёме работ порядок и стоимость их выполнения определяются отдельным соглашением.

7. Обновления и информирование пользователей

7.1 Порядок применения обновлений

Размещение на мощностях правообладателя (интернет-сервис). Обновление применяется силами ООО «ИНТЕБРИКС» в согласованное с заказчиком технологическое окно.

Установка в контуре заказчика (on-premise) и в изолированном контуре (air-gap). Обновление выполняется штатным скриптом `scripts/update.sh`, входящим в состав поставки. Последовательность операций скрипта:

1. создание резервной копии (логическая выгрузка базы данных, журналы, конфигурация) — копия используется как точка отката;
2. получение новых образов контейнеров из переданного правообладателем архива обновления (`--air-gap images.tar`) — единственного источника образов для поставки;
3. остановка компонентов API и фоновой обработчика; хранилище данных продолжает работу;

4. применение миграций базы данных (`alembic upgrade head`) с ожиданием готовности API по `/health/live`;
5. пересоздание компонентов на новых образах;
6. контрольная проверка работоспособности (`/health/live`, до 90 секунд с повторными попытками);
7. при отрицательном результате любого из шагов 2–6 — **автоматический откат** к резервной копии, созданной на шаге 1, посредством `scripts/restore.sh`, с записью в журнал `logs/update.log`. Автоматический откат может быть отключён параметром `--no-rollback` для последующего разбора инцидента.

Ход обновления протоколируется в `logs/update.log` с отметками времени в UTC.

Поставка обновления в изолированный контур. Формируется архив `intebrix-airgap-<версия>.tar.gz` (образы контейнеров, файл описания развёртывания, шаблон конфигурации, эксплуатационные скрипты, контрольные суммы SHA-256). Архив передаётся заказчику согласованным способом (раздел 3.9); целостность проверяется по контрольным суммам перед установкой.

7.2 Виды обновлений

Вид	Содержание	Порядок выпуска
Функциональное (минорная версия)	новая функциональность с сохранением обратной совместимости	плановый
Исправляющее (патч-версия)	устранение дефектов	плановый или внеочередной
Исправление безопасности	устранение уязвимостей, в том числе в сторонних компонентах	внеочередной, с приоритетом
Мажорное	изменения, требующие действий на стороне заказчика	плановый, с предварительным уведомлением

7.3 Информирование пользователей

О каждом выпуске пользователям, находящимся на сопровождении, направляется уведомление о выпуске, содержащее:

- номер и дату версии;
- краткое описание выпуска;
- перечень новой функциональности;
- перечень изменений, нарушающих обратную совместимость, либо указание на их отсутствие;
- перечень известных ограничений;

- порядок и сроки применения обновления для размещения на мощностях правообладателя и для установки в контуре заказчика.

Полный журнал изменений по всем версиям ведётся в файле `docs/CHANGELOG.md` репозитория и публикуется в открытом разделе сайта правообладателя по адресу <https://intebrix.ru/changelog> (на русском языке); в комплект закрытого контура входит памятка `README.md` с номером версии и датой сборки.

Об изменениях, требующих действий на стороне заказчика (изменение формата данных, изменение программного интерфейса, изменение требований к окружению), уведомление направляется заблаговременно, до выпуска версии.

О мажорных и несовместимых изменениях заказчики уведомляются не позднее чем за 30 календарных дней до выпуска. Каналы уведомления: адрес электронной почты, указанный в договоре, и журнал выпусков на сайте правообладателя.

8. Персонал

8.1 Состав работ, требующих квалифицированного персонала

Роль	Область ответственности
Разработчик серверной части	Python 3.12, FastAPI, Pydantic, SQLAlchemy (асинхронный режим), Celery, PostgreSQL, миграции Alembic; проектирование и реализация программного интерфейса
Разработчик расчётных моделей (специалист по методам оптимизации)	линейное и смешанно-целочисленное программирование, задачи маршрутизации (VRP/VRPTW) и календарного планирования (APS); решатели HiGHS и CBC, библиотека <code>python-mip</code> ; <code>NetworkX</code> , <code>NumPy</code> , <code>SciPy</code> ; постановка и настройка оптимизационных моделей
Разработчик клиентской части	React 19, TypeScript, Vite; проектирование интерфейсов работы с производственными и логистическими данными; требования доступности интерфейса
Инженер по развёртыванию и эксплуатации	Docker, Docker Compose, nginx, PostgreSQL, Valkey; сборка и поставка дистрибутивов, в том

Роль	Область ответственности
Инженер по тестированию	числе для изолированного контура; резервное копирование и восстановление; наблюдение за состоянием установок pytest, Playwright, k6; проектирование регрессионных и сквозных сценариев; проверка корректности и воспроизводимости результатов расчёта
Специалист технической поддержки	приём и разбор обращений, работа с журналами платформы, консультирование по эксплуатации

Одна роль может исполняться одним специалистом, и один специалист может совмещать несколько ролей.

8.2 Требования к квалификации

- высшее образование в области информационных технологий, прикладной математики, системного анализа или смежных областях;
- для разработчиков — практический опыт промышленной разработки на соответствующем стеке от трёх лет;
- для разработчика расчётных моделей — подготовка в области исследования операций и математического программирования, практический опыт постановки и решения задач оптимизации производственного и транспортного планирования;
- для инженера по развёртыванию — практический опыт эксплуатации контейнеризованных приложений, в том числе в закрытых контурах без доступа в сеть Интернет;
- знание русского языка на уровне, достаточном для ведения технической документации и взаимодействия с заказчиками.

8.3 Фактическая численность

Совершенствование программы, устранение сбойных ситуаций, техническую поддержку и гарантийное обслуживание обеспечивает правообладатель собственными силами — привлечёнными специалистами, гражданами Российской Федерации, по гражданско-правовым договорам. Работников по трудовым договорам у правообладателя нет; сторонние организации к сопровождению не привлекаются.

Направление работ	Численность, чел.	Режим занятости	Форма привлечения	Гражданство
Руководство разработкой, архитектура, расчётные	1	постоянно	учредитель общества, автор программы;	Российская Федерация

Направление работ	Численность, чел.	Режим занятости	Форма привлечения	Гражданство
ядра, программный интерфейс, техническая поддержка и гарантийное обслуживание			гражданско-правовой договор с автором	
Инженерная дисциплина, испытания и внедрение доработок, установка и обновление у заказчиков	1	при наличии работ	внештатный специалист, гражданско-правовой договор	Российская Федерация
Работа с заказчиками, сопровождение внедрений, модели данных	1	при наличии работ	внештатный специалист, гражданско-правовой договор	Российская Федерация

Общая численность персонала, обеспечивающего поддержание жизненного цикла программы, — 3 человека (все — граждане Российской Федерации, внештатные): один специалист занят постоянно, два привлекаются по гражданско-правовым договорам при наличии работ по совершенствованию или сопровождению. Все трое участвуют и в модернизации программы, и в её сопровождении. Квалификация специалистов соответствует разделу 8.2; договоры предоставляются по запросу.

8.4 Фактические адреса размещения

Объект	Фактический адрес
Инфраструктура разработки: рабочая станция сборки и хранения исходного и объектного кода	127051, г. Москва, ул. Петровка, д. 20/1, помещ. 17
Инфраструктура разработки: удалённая копия репозитория (сервис GitFlic)	серверные мощности сервиса — на территории Российской Федерации; оператор сервиса — ООО «РеСолют», г. Москва, Огородный проезд, д. 16/1, стр. 5 (раздел 2.2)
Разработчики	г. Москва, Российская Федерация (работа выполняется удалённо, с

Объект	Фактический адрес
Служба технической поддержки	территории Российской Федерации) 127051, г. Москва, ул. Петровка, д. 20/1, помещ. 17
Работы по разработке, тестированию, выпуску версий, технической поддержке и сопровождению выполняются только на территории Российской Федерации.	

9. Контакты службы поддержки

Канал	Реквизит
Электронная почта службы поддержки (приём обращений, консультации)	support@intebrix.ru
Электронная почта по общим вопросам	info@intebrix.ru
Телефон	+7 916 925-54-71
Форма обратной связи	https://intebrix.ru/contacts
Почтовый адрес	127051, г. Москва, вн.тер.г. муниципальный округ Тверской, ул. Петровка, д. 20/1, помещ. 17
Наименование организации	ООО «ИНТЕБРИКС», ИНН 9707059284, ОГРН 1267700207895

Язык обращения и ответа — русский.

Для обращений в службу технической поддержки выделен адрес support@intebrix.ru (решение правообладателя от 19.08.2026); общий адрес info@intebrix.ru сохраняется для прочих вопросов. Приём обращений по телефону +7 916 925-54-71 осуществляется в рабочие дни с 10:00 до 19:00 по московскому времени; обращение, принятое по телефону, регистрируется в том же порядке, что и письменное, и подтверждается заявителю по электронной почте.

Режим работы службы поддержки: рабочие дни, с 10:00 до 19:00 по московскому времени. Обращение по критическому инциденту, поступившее вне режима работы по электронной почте или через форму обратной связи, регистрируется автоматически; отсчёт срока реакции начинается с ближайшего рабочего часа. Расширенный режим приёма обращений предоставляется по отдельному соглашению о сопровождении.

10. Сопутствующие документы

Документ	Назначение
Описание функциональных	состав и назначение расчётных

Документ	Назначение
характеристик программного обеспечения	ядер, решаемые задачи, ограничения
Инструкция по установке программного обеспечения	требования к окружению, порядок установки и первичной настройки
Документация по эксплуатации программного обеспечения	руководство пользователя и администратора
Настоящее описание процессов жизненного цикла	организация разработки и сопровождения
Протокол испытаний на российских операционных системах	результаты установки и проверок на Astra Linux Special Edition 1.8.2 и РЕД ОС 8.0.2
Письмо о технических средствах хранения и компиляции кода и о технических средствах управления лицензионными ключами	место нахождения технических средств, средства сборки, лицензирование

Внутренние источники, на которых основано настоящее описание, входят в состав исходного дистрибутива (репозитория): `docs/CHANGELOG.md` (журнал изменений), `docs/deployment-guide.md` (руководство по развёртыванию), `docs/user-guide-ru.md` (руководство пользователя), `docs/api-reference.md` (описание программного интерфейса), `docs/troubleshooting.md` (типовые неисправности и порядок их диагностики), `docs/architecture.md` (архитектура), `THIRD-PARTY-NOTICES.txt` (состав сторонних компонентов и их лицензии). В комплект закрытого контура из перечисленного входят `THIRD-PARTY-NOTICES.txt`, файл `VERSION` и памятка `README.md`; журнал изменений публикуется на сайте правообладателя (раздел 7.3).
