

Подробный паспорт Двигателя

Тимон Ядро · Архитектор и разработчик интеграции

Редакция 2026-09-15 · Редакция 15 сентября 2026 года; состояние функций уточняется документом готовности

Область применимости и актуальные ограничения:
<https://team-on.ru/yadro/docs/readiness.html>

Оглавление

Область применения источника

1. Цель и смысл
 2. Пользователи и ценность
 3. Системная карта
 4. Ответственность и данные
 5. Архитектурные блоки и код
 6. Контракты и жизненный цикл
 7. Полномочия и границы доверия
 8. Ошибки и наблюдаемость
 9. Развёртывание и сопровождение
 10. Что есть и что предстоит
 11. Приёмка
 12. Открытые решения
- Текущий слой контекста

Область применения источника

Двигатель — серверная часть Core, существующие каналы, история, задания, полномочия и инструменты. Описанные далее обязанности задают архитектуру; фактическая включённость функций зависит от версии экземпляра. Свежий код не считается автоматически развёрнутым на публичном сервере.

Управляемый Context в проверенном срезе является выключенным по умолчанию лабораторным режимом владельца. Общая корпоративная история всех интерфейсов и файловая изоляция многопользовательских исполнителей не считаются завершёнными.

1. Цель и смысл

Превратить запрос человека в управляемое выполнение: сохранить задачу и контекст, подключить разрешённую модель и инструменты, довести действие до наблюдаемого результата. Пользователь должен продолжить ту же работу после переключения между Штатом, Шлюзом и подключённым каналом. Оператор должен видеть, что действительно произошло, в том числе когда результат внешнего действия неизвестен.

Двигатель — владелец агентной работы. Core управляет состоянием и полномочиями; OpenCode исполняет модельный цикл. Наличие OpenCode не означает, что вся бизнес-логика переносится в него или что собственная языковая модель входит в продукт.

2. Пользователи и ценность

Сотрудник получает результат и историю; оператор — управляемые операции и диагностику; администратор — обслуживаемый экземпляр с понятным состоянием. Разработчик интеграции получает проверяемые контракты вместо доступа к БД. Критерии ценности: задача завершается результатом, история восстанавливается, отмена работает, повтор не создаёт лишний внешний эффект, чужие данные недоступны. Процент успешных задач и задержки измеряются на стенде; целевые числа ещё не заданы.

3. Системная карта

```
flowchart LR
    Inputs[Штат / Штурвал / Штрих / Шлюз / каналы] --> API[Проверка actor и входного контракта]
    API --> Core[Core: задача и состояние]
    Core --> Queue[Очередь и жизненный цикл]
    Queue --> Executor[OpenCode]
    Executor --> Model[Разрешённая модель]
    Executor --> Tools[Инструменты и коннекторы]
    Tools --> Result[Результат и квитанция]
    Result --> Core
    Core --> Store[История / context / файлы]
    Core --> Events[События и чтение состояния]
```

Это целевая корпоративная карта. Разные текущие входы имеют разные механизмы identity; единый scored web-вход ещё требуется реализовать.

4. Ответственность и данные

Core владеет пользователем/ролью в своём runtime, агентом, диалогом, состоянием задания, назначенным контекстом, расписаниями и receipt. SQLite и рабочие области составляют operational state. Web-клиенты читают разрешённые представления и подают команды, а не изменяют таблицы напрямую. Один логический writer выполняет составные изменения; исторические пути требуют сверки с ADR-016.

OpenCode владеет исполнением модельного turn и техническими данными своей сессии; Core сохраняет связь с бизнес-задачей и её состоянием. Шлюз владеет локальным исполнением device command. Нельзя объявлять локальное действие успешным по факту его постановки в серверную очередь.

5. Архитектурные блоки и код

Блок	Источник	Роль
Runtime и маршрутизация	src/main.ts, provider-dispatch.ts, agent-opencode.ts	Выбранный исполнитель и агентный цикл
Сериализация и жизненный цикл	session-turn-queue.ts, runtime-turn-context.ts, turn-receipt.ts	Продолжение, порядок, результаты
Данные	db.ts, reminder-store.ts	История, события, напоминания
Команды управления	core-command.ts, operator-runtime.ts	Typed commands и Core authority
Устройства и каналы	edge-store.ts, channel-edge.ts, bridge-compatible.ts	Pairing, actor/agent binding, delivery
Вложения и речь	bridge-attachments.ts, bridge-voice.ts, audio-chunks.ts	Рабочая область, лимиты, обработка медиа
Диагностика	main-dashboard.ts, monitor.ts	Существующая техническая поверхность

Пути относительно src/ в этом репозитории. Это карта проверенных точек интеграции, не утверждение, что каждая ветка старого runtime уже соответствует целевому профилю.

6. Контракты и жизненный цикл

Вход должен связывать authenticated actor, agent, разговор/рабочую область, request ID, текст/вложения и разрешённые действия. Идентификаторы не выводятся из display name. Конкретные существующие schemas берутся из модулей Core; не добавляем несовместимые поля без новой версии контракта.

Целевой переход: принято → выполняется → ожидает данных/подтверждения → завершено, ошибка либо отменено. Неизвестный внешний эффект требует status/reconcile. Доставка события может повторяться, но повтор command ID не должен повторять бизнес-действие. Переключение провайдера после возможного эффекта не означает безопасного повторного запуска. Подтверждение изменения связано с digest/revision.

7. Полномочия и границы доверия

Контекст модели и содержимое файла не являются разрешением. Сервер определяет actor и область доступа, повторно проверяет их при исполнении после очереди. Отдельно контролируются агентные shell/file/network инструменты: фильтрация UI не ограничивает процесс исполнителя. Ключ модели не должен становиться доступным произвольным child tools; способ доставки credentials требует испытания.

Существующий company-scope Operator не является корпоративным RBAC. Требуются проектные/пользовательские scopes, привязка IdP и отзыв действующих полномочий. Пути, файлы и секреты защищаются независимо от роли owner.

8. Ошибки и наблюдаемость

Модель недоступна: явная ошибка readiness; запрещён незаявленный внешний fallback. Инструмент завершился неизвестно: сохраняем исходное намерение и выясняем эффект. Занята рабочая область: последовательная обработка или явный конфликт. Диск заполнен: контролируемая ошибка без ложного «файл создан». Отмена: проверяется остановка текущей работы и дальнейших побочных действий.

Наблюдаемость: request/run/receipt IDs, очередь, длительность этапов, provider readiness, ошибка и безопасный контекст для поддержки. Не журналируем raw secrets, а содержание разговоров защищаем политикой доступа/хранения.

9. Развёртывание и сопровождение

Core/OpenCode на сервере заказчика или тестовой VM; технические порты закрыты от браузера. Версии, native dependencies и модель закреплены. Данные отделены от immutable release. Backup охватывает согласованный набор БД/файлов/настроек; миграция и rollback проверяются совместно. Single-node не равен HA.

10. Что есть и что предстоит

Есть исходный Core 0.18.1, локальные unit/build и технический smoke, OpenCode startup, операторские команды и bridge. Не приняты: Linux-поставка, российская модель с tools, реальная STT, полный web identity/RBAC, новая UI/клиентская связка. Локальный synthetic recording test не является измерением качества речи.

11. Приёмка

ENT-02/03/04/08/15/16/18/19/20/21/36/38. Минимум: чистая установка; task с tools и файлом; restart и resume; дубль/отмена/неизвестный эффект; отказ чужому actor; backup/restore. DEMO-01/02/03/07/08. В отчёте — точные версии и реальные результаты.

12. Открытые решения

Выбранная модель/STT, схема делегированной web identity, Linux/BOM, лимиты, лицензирование и egress. Их отсутствие явно блокирует соответствующий ENT, но не переносит ответственность на пользовательский интерфейс.

Общий контекст: системная карта (см. оглавление документации), критерии приёмки (см. оглавление документации), план демо (см. оглавление документации).

Текущий слой контекста

Встроенный TeamON MCP предоставляет логические области компании, агента и пользователя. Контекстные документы, подтверждённая память и состояние работы имеют разных владельцев. Поиск публичной документации сайта является отдельным сервисным корпусом и не заменяет эти функции.

Управляемый Context описывает объекты и коллекции с проверками областей доступа. Включение допускается в лабораторном профиле владельца; выдавать этот режим за готовую общекорпоративную базу знаний нельзя. До выпуска необходимо подтвердить изоляцию процессов, чтение исходников и версий, отзыв прав и обходные файловые пути.

Основание документа

Сверка продуктовых документов и закреплённых исходников 15 сентября 2026 года. Проверка документации не заменяет промышленную приёмку. Полная применимость определяется документом о состоянии компонентов.