RDV

RDV MI FAQ

Популярные вопросы и ответы на них

Продуктово-архитектурные вопросы

- Чем RDV MI отличается от обычных самописных обменов в 1С?
- RDV MI — это замена RabbitMQ, ESB или iPaaS?
- Чем наши подходы отличаются от КД2, КД3, Шины данных
- Как RDV MI понимает, какие данные нужно отправить?
- Как RDV MI помогает не потерять сообщения?
- Что такое mi-monitor и зачем он нужен?
- Чем мониторинг RDV MI отличается от обычной Grafana с графиками?
- Зачем в RDV MI нужен АРМ сверки данных, если уже есть мониторинг?
- Как работает АРМ сверки данных?
- В чем главная ценность RDV MI как единого продукта?

Технические вопросы

- RDV MI — это внешняя обработка, расширение или часть конфигурации?
- Почему RDV MI не делает ставку на внешние обработки?
- Как в RDV MI описывается формат сообщения?
- Что значит «RDV MI сам понимает свой формат»?
- А если внешняя система требует свой формат JSON?
- RDV MI — это low-code?
- Как RDV MI регистрирует изменения к обмену?
- Почему «один объект — одно сообщение», а не один большой пакет?
- Как RDV MI обеспечивает последовательность обмена?
- Какие транспорты поддерживает RDV MI?
- Что нужно изменить, если появился новый получатель данных?
- Почему RDV MI не ESB?
- Где писать прикладную логику конвертации?
- Как RDV MI помогает ускорять обработку больших объемов?
- Как RDV MI проверяет, что данные действительно доехали корректно?
- Как RDV MI связан с мониторингом?
- Можно ли использовать RDV MI только для простого HTTP-обмена без RabbitMQ?
- Насколько RDV MI открыт для доработок?

Продуктово-архитектурные вопросы

Чем RDV MI отличается от обычных самописных обменов в 1С?

Самописный обмен обычно решает одну конкретную задачу: «отправить заказ», «получить остатки», «дернуть API». Но вокруг этого быстро появляется куча одинаковых технических проблем:

- Как понять, что именно было отправлено
- Как повторить отправку
- Где смотреть ошибку
- Как отследить, дошло ли сообщение
- Как понять, совпали ли данные после обмена
- Как сопровождать интеграцию через полгода
- Как новому разработчику разобраться, что вообще происходит

- **RDV MI стандартизирует эти вещи**

Вместо набора уникальных обработок «каждый разработчик сделал как понял»
появляется единый подход

- Единые правила регистрации изменений
- Единые механизмы формирования сообщений
- Единые транспорты
- Единые журналы
- Единый мониторинг
- Единые сценарии диагностики
- Единая сверка данных

То есть RDV MI отличается не тем, что «тоже умеет отправлять JSON». JSON все умеют отправлять, великая магия XXI века.
Отличие в том, что **RDV MI делает интеграцию управляемой и сопровождаемой**

RDV MI — это замена RabbitMQ, ESB или iPaaS?

Нет, RDV MI не заменяет RabbitMQ
RabbitMQ — это транспорт и брокер сообщений. RDV-MI использует его как один из вариантов доставки

RDV MI не является классической ESB для всего предприятия
Он не пытается стать центральной шиной для всех систем в мире

RDV MI — это 1С-центричный интеграционный слой, который решает главную боль 1С-проектов: как системно строить обмены между 1С и внешними системами, не превращая каждую интеграцию в отдельное произведение народного творчества

Чем подход RDV MI отличается от КД2, КД3, Шины данных?

1 Разработка

Разработка правил конвертации

В RDV MI присутствуют встроенные механизмы для разработки правил конвертации. Общие функции формирования сообщений обмена уже встроены в решение

Объем кода для старта

При разработке правил конвертации между идентичными или похожими конфигурациями требуется минимум кода

Простота разработки

В поставку RDV MI входит набор служебных процедур и функций, которые облегчают описание прикладного кода конвертации

Отладка

Весь код хранится в конфигурации, 1 объект = 1 сообщение. Отладка выполняется средствами платформы 1С, не требует дополнительных инструментов

Версионирование / доставка правил до продуктива

Разработка правил конвертации ведется через конфигуратор, есть возможность подключить Git для хранения истории изменений и организовать поставку правил в продуктив через релизы

Зависимость от формата

Используется формат JSON с набором предопределенных полей. При необходимости формат можно изменить

Транспорт

Используется http / брокер RabbitMQ. Есть возможность добавления нового транспорта обмена

Критерий	Модуль интеграции	КД2	КД3 + EnterpriseData	Шина данных
Разработка правил конвертации	Встроенные механизмы (как в КД3, НО В коде)	Визуальные правила	Визуальные правила + код	Нет
Объем кода для старта	Минимальный при совпадении структур	Не нужно описывать правила	Зависит от EnterpriseData	Значительный
Простота разработки	Обычный код + вспомогательные Функции	Визуальная модель	Сложная модель + ED	Разнесено по системам
Отладка	Стандартный отладчик 1С	Сложная трассировка	Еще сложнее	Через несколько систем
Версионирование / доставка правил до продуктива	Полностью через Git / через релизы	Нет	Нет	Частично
Зависимость от формата	Гибкий (свой)	Свой	EnterpriseData	Зависит
Транспорт	HTTP / RabbitMQ / расширяемо	HTTP / файл / FTP	HTTP / файл / FTP	HTTP / RabbitMQ / расширяемо

2 Поддержка

Инфраструктура

При обмене через http разворачивание и поддержка дополнительной инфраструктуры не требуется, при обмене через RabbitMQ требуется выделенный сервер с минимальными требованиями

Отказоустойчивость

Данные обрабатываются в независимых транзакциях. Ошибка обработки сообщения не приводит к полной остановке обмена

Контроль целостности данных

В RDV MI встроен механизм обмена хешами объектов, которые участвуют в интеграции. Можно сверить целостность данных на стороне источника и/или приемника

Производительность

Скорость обмена данными обеспечивается за счет «легкого» формата сообщений обмена (JSON), использования http-сервисов или брокера сообщений, встроенные возможности организовать параллельную обработку данных

Мониторинг и логирование

В RDV MI встроены механизмы логирования и регистрации ошибок. Есть возможность использовать совместимый продукт RDV MI Monitor на базе ClickHouse + Grafana для мониторинга инфраструктуры обменов.

Стоимость

RDV MI, RabbitMQ, RDV MI Monitor являются полностью бесплатными и не требуют лицензий

Критерий	Модуль интеграции	КД2	КДЗ + EnterpriseData	Шина данных
Инфраструктура	Не требуется (брокер опционален)	Не требуется	Не требуется	Требуется
Отказоустойчивость	1 объект = 1 сообщение	Откат всей загрузки	Часто транзакционная	Опционально
Контроль целостности данных	Хеширование	Нет	Нет	Нет
Производительность	Легкий формат + поток	HML	HML + ED	Зависит
Мониторинг и логирование	Встроенные механизмы	Ограничены	Ограничены	Частично реализовано в Шине
Стоимость	Бесплатно	Бесплатно	Бесплатно	Лицензии

3 Масштабируемость

Разделение разработки*

Разработка правил выгрузки и загрузки выполняется раздельно. Может разрабатываться разными командами

Подключение новых источников

Для подключения новых источников данных в инфраструктуру обмена не требуются доработки на стороне приемника

Зависимость от типовых релизов конфигураций 1С

При обновлении конфигурации, участвующей в обмене на новый типовой релиз, не требуется доработка интеграции, если не изменился контракт данных (структура и бизнес-смысл передаваемой информации)

Обработка больших объемов

В RDV MI есть возможность потоковой (параллельной) обработки загружаемых данных

* выгрузка и загрузка могут разрабатываться отдельными командами

Критерий	Модуль интеграции	КД2	КДЗ + EnterpriseData	Шина данных
Разделение разработки	Полное (выгрузка/загрузка)	Нет	Нет	Частично
Подключение новых источников	Без изменений приемника	Доработка правил	Доработка правил	Через шину
Зависимость от типовых релизов конфигураций 1С	Нет	Нет	Есть	Нет
Обработка больших объемов	Потоковая + параллельность	Ограничена	Ограничена	Да

Как RDV MI понимает, какие данные нужно отправить?

В RDV MI есть механизм регистрации изменений

Когда в 1С изменяется объект — например, заказ, контрагент, номенклатура, остатки или документ — система может зарегистрировать это изменение к обмену. После этого RDV MI формирует сообщение по заданным правилам и отправляет его получателю

Как RDV MI помогает не потерять сообщения?

RDV MI строится вокруг идеи, что сообщение должно быть не «вызовом функции»,
а самостоятельной сущностью, которую можно отследить

У сообщения есть жизненный цикл:

- создано
- подготовлено
- отправлено
- получено
- обработано
- завершено / завершено с ошибкой

За счет журналов, контекста сообщений, очередей обработки и мониторинга можно понять:

- Какое сообщение было сформировано
- Когда оно было отправлено
- Куда оно должно было попасть
- Было ли оно получено
- Возникла ли ошибка обработки
- Можно ли выполнить повторную обработку
- Какие сообщения «зависли» или не дошли

Что такое mi monitor и зачем он нужен?

Mi monitor — это мониторинговый контур RDV MI на базе ClickHouse и Grafana
Он нужен, чтобы видеть состояние обменов не внутри отдельной базы 1С, а централизованно

Mi-monitor показывает:

- Сколько сообщений отправлено
- Сколько сообщений получено
- Какие базы онлайн или офлайн
- Где растут очереди
- Какие сообщения не были доставлены
- Сколько времени занимает доставка
- Какие типы сообщений создают наибольшую нагрузку
- Что находится внутри конкретного JSON-сообщения

То есть RDV MI отвечает за выполнение обмена, а mi-monitor — за наблюдаемость

Без мониторинга интеграция часто работает по принципу: «Пока клиент не написал в чат, что все сломалось, значит все хорошо»

Mi-monitor делает иначе: **проблемы можно видеть заранее** — по очередям, задержкам, недоставленным сообщениям и расхождениям между отправкой и получением

Чем мониторинг RDV MI отличается от обычной Grafana с графиками?

Обычная Grafana сама по себе ничего не знает об интеграциях. Она просто рисует то, что ей дали

Ценность mi-monitor в том, что он построен вокруг предметной модели RDV MI

база-источник | база-получатель | вид сообщения | идентификатор сообщения | статус доставки | время отправки | время получения | очередь | JSON сообщения | недоставленные сообщения

То есть это не абстрактные графики CPU, памяти и «количество чего-то там».
Это дашборды именно про обмены:

- Какие сообщения ушли, но не дошли
- Какая база перестала отвечать
- Какой тип данных доставляется дольше всего
- По какому направлению накопилась очередь
- Какой JSON реально был отправлен

Именно поэтому mi monitor — это не просто визуализация. Это инструмент диагностики интеграций

Зачем в RDV MI нужен APM сверки данных, если уже есть мониторинг?

Потому что техническая доставка сообщения не гарантирует, что бизнес-данные совпали

Сообщение может быть: отправлено | получено | обработано | без технических ошибок

Но при этом во внешней системе данные могут оказаться другими

- Не тот статус
- Не та сумма
- Не тот склад
- Потерялась характеристика
- Округлилось количество
- Неправильно сопоставилась номенклатура
- Документ создался, но не в том состоянии

Mi monitor отвечает на вопрос **«Сообщение дошло?»**

АРМ сверки данных отвечает на другой вопрос: **«Данные в системах реально совпадают?»**

Это принципиально разные уровни контроля. Поэтому АРМ сверки — это третий слой продукта:

RDV MI	Mi monitor	АРМ сверки
обмен данными	техническая наблюдаемость	контроль фактической корректности данных

Вместе они закрывают не только доставку, но и качество интеграции

Как работает АРМ сверки данных?

АРМ позволяет загрузить данные из разных источников, настроить правила сравнения и получить отчет о расхождениях

Типовой сценарий

1. Пользователь выбирает два набора данных
2. Первый набор загружается, например, из текущей базы 1С
3. Второй набор загружается из ClickHouse, Excel, другой базы или внешней системы
4. Пользователь настраивает ключи сопоставления
5. Указывает, какие колонки нужно сравнивать
6. Указывает, какие колонки нужно только вывести в отчет
7. При необходимости добавляет вычисляемые поля для нормализации данных
8. Выполняет сравнение
9. Получает результат: совпадает, отличается, есть только в первой системе, есть только во второй системе, ошибка ключа

Главная сила АРМа — в гибкости

Он не требует, чтобы данные в разных системах были идеально одинаковой структуры. Можно добавлять вычисляемые колонки, собирать ключи из нескольких полей, приводить значения к сопоставимому виду и сравнивать уже нормализованные данные.

Это важно, потому что реальные интеграции почти никогда не выглядят как учебный пример «поле А равно полю Б». Обычно там «три реквизита в одной системе соответствуют одному статусу во второй, но только если документ проведен и дата больше прошлого вторника». Классика жанра

В чем главная ценность RDV MI как единого продукта?

Главная ценность RDV MI в том, что он закрывает весь контур интеграции, а не один отдельный кусок

Многие решения закрывают только часть задачи

- **RabbitMQ** доставляет сообщения
- **HTTP API** передает данные
- **Самописная обработка** формирует JSON
- **Grafana** рисует графики
- **Excel** помогает вручную сравнить выгрузки

RDV MI объединяет эти части **в единую модель**

разработка обменов | регистрация изменений | формирование сообщений | доставка | обработка | журналирование | мониторинг | диагностика | сверка данных | сопровождение

То есть продукт отвечает не только на вопрос: «Как отправить данные?»
А на более важные вопросы:

- Что именно изменилось?
- Что было отправлено?
- Куда оно должно было попасть?
- Дошло ли оно?
- Было ли обработано?
- Где ошибка?
- Совпали ли данные после обмена?
- Что делать поддержке, если не совпали?

Именно это отличает RDV MI от набора технических инструментов. Это не просто механизм обмена. **Это методологически оформленный контур управления интеграциями для 1С-проектов**

Технические вопросы

RDV MI — это внешняя обработка, расширение или часть конфигурации?

RDV MI может поставляться как расширение или встраиваться в конфигурацию через сравнение/объединение

Принципиальная идея продукта: код интеграции должен жить в конфигурации или расширении, а не в наборе внешних обработок и файлов рядом с базой

Это дает **нормальный промышленный контур разработки**

- Код версионируется в Git;
- Изменения проходят стандартный процесс разработки и тестирования
- Поставка на тест и прод выполняется централизованно
- Прикладная логика не теряется между средами
- Разработчик работает в привычном контуре 1С
- Отладка выполняется штатными инструментами

RDV MI не заменяет разработку «настройками в интерфейсе», а дает стандартный каркас, в который разработчик помещает прикладную логику интеграции

Почему RDV MI не делает ставку на внешние обработки?

Внешние обработки удобны для быстрых разовых задач, но плохо подходят как основа промышленного интеграционного контура

Типовые проблемы такого подхода

- На тесте одна версия обработки, на проде другая
- Часть логики хранится локально у разработчика
- Изменения вносятся вручную и плохо отслеживаются
- Сложно понять, какая версия актуальна
- Сложнее выстроить Git, code review, тестирование и поставку
- Интеграция превращается в набор разрозненных файлов

RDV MI исходит из другой логики

Интеграция — это часть прикладной архитектуры системы. Значит, ее код должен версионироваться, тестироваться и поставляться так же, как остальной код конфигурации

Как в RDV MI описывается формат сообщения?

RDV MI использует стандартный JSON-формат, разделенный на два смысловых блока

1. data — данные объекта
2. meta — служебная информация сообщения

В data передаются реквизиты объекта, значения полей, ссылки, GUID и другие данные, необходимые для загрузки

В meta хранится технический контекст

- Идентификатор сообщения
- База-источник
- Тип объекта
- Время формирования
- Служебные признаки для маршрутизации, диагностики и мониторинга.

За счет этого сообщение содержит не только бизнес-данные, но и **технический контекст, необходимый для сопровождения обмена**

Что значит «RDV MI сам понимает свой формат»?

Если обмен идет между системами, где с обеих сторон используется RDV MI, разработчику не нужно каждый раз вручную описывать очевидную техническую механику

RDV MI уже понимает

- Где в сообщении бизнес-данные
- Где служебный контекст
- Как работать со ссылками
- Как передавать GUID
- Как интерпретировать тип объекта
- Как связать сообщение с мониторингом и журналами

Разработчик описывает, какие данные нужно выгрузить и как их обработать, а типовая техническая часть выполняется платформой

Коротко

Все, что может работать автоматически, должно работать автоматически.
Разработчик должен заниматься бизнес-логикой, а не заново писать одинаковую упаковку JSON для каждого обмена

А если внешняя система требует свой формат JSON?

RDV MI не заставляет все внешние системы принимать его внутренний формат

Если внешняя система требует собственную структуру JSON, другие имена полей, плоский формат, вложенные блоки или специфические правила API — формат можно адаптировать прикладным кодом

Возможны два основных сценария

RDV MI ↔ RDV MI
используется родной формат RDV MI

RDV MI ↔ внешнее API
формат адаптируется под требования внешней системы

При этом базовый формат уже существует, и его не нужно изобретать с нуля для каждого обмена. Если внешний формат требует отклонений — дорабатывается только специфичная часть

RDV MI — это low-code?

Нет. RDV MI не является low-code-платформой и не обещает, что сложную интеграцию можно собрать мышкой без разработчика

Это сознательная позиция

Сложная интеграция почти всегда требует кода

- Нестандартные правила сопоставления
- Сложные условия выгрузки
- Особая логика загрузки
- Проверки перед записью
- Обработка ошибок
- Преобразование данных под модель внешней системы

RDV MI снижает порог входа не за счет «магических настроек», а за счет автоматизации типовой технической рутины

- Регистрация изменений
- Формирование сообщений
- Сериализация типовых значений
- Работа с GUID и ссылками
- Транспорт
- Очереди
- Журналы
- Мониторинг
- Контроль целостности

Простые вещи должны работать автоматически. **Сложная бизнес-логика должна писаться кодом, который можно отлаживать, тестировать и версионировать**

Как RDV MI регистрирует изменения к обмену?

RDV MI использует механизм регистрации изменений, при котором изменение объекта фиксируется как отдельная запись для последующей обработки

Модель выглядит так

- изменился объект в 1С
- → изменение зарегистрировано к обмену
- → сформировано сообщение
- → сообщение отправлено
- → сообщение получено и обработано
- → результат доступен в журналах и мониторинге

Одна регистрация соответствует конкретному объекту, направлению обмена и будущему сообщению

Такой подход дает понятный жизненный цикл обмена

- Можно увидеть, какой объект был зарегистрирован
- Понять, какое сообщение из него сформировалось
- Отследить отправку и получение
- Повторить обработку
- Локализовать ошибку на уровне конкретного объекта

Почему «один объект — одно сообщение», а не один большой пакет?

- Пакетная выгрузка удобна **до первой ошибки**

Выгружается **1 000** объектов → **999** корректные → **1** объект содержит ошибку →

Весь пакет зависает или требует ручного разбора

RDV MI использует более атомарный подход

один объект = одно сообщение = один элемент очереди

Это делает интеграцию более прозрачной и сопровождаемой

- Ошибка одного объекта не блокирует остальные
- Каждое сообщение можно отдельно отследить
- Проще повторить обработку
- Проще найти проблемный объект
- Проще анализировать очередь
- Проще подключать мониторинг
- Проще масштабировать доставку через брокер

Как RDV MI обеспечивает последовательность обмена?

Для многих интеграций важен порядок доставки и обработки данных

Например

Сначала должен прийти контрагент → Потом договор → Потом документ, который ссылается на этот договор

RDV MI позволяет управлять порядком обработки через регистрацию изменений, очереди и временные метки

Это особенно важно для ссылочных данных, документов и зависимых объектов. Если порядок нарушить, приемник может получить битые ссылки, ошибки поиска или некорректное состояние данных

Задача RDV MI — не просто отправить набор сообщений, а **сохранить управляемый жизненный цикл обработки**

Какие транспорты поддерживает RDV MI?

Основные варианты транспорта:

- RabbitMQ
- HTTP

→ Выбор зависит от сценария

Сценарий	Рекомендуемый транспорт
Простая интеграция двух систем	HTTP
Временный или небольшой обмен	HTTP
Несколько получателей	RabbitMQ
Событийная архитектура	RabbitMQ
Развивающийся интеграционный контур	RabbitMQ
Нужна маршрутизация без доработки источника	RabbitMQ

HTTP подходит **для простых сценариев**

RabbitMQ лучше подходит **для масштабируемого интеграционного контура,** где появляются новые получатели, разные типы сообщений, очереди, маршрутизация и независимая обработка

Что нужно изменить, если появился новый получатель данных?

При использовании RabbitMQ источник не должен знать всех получателей

Если появляется новая система-получатель, сценарий выглядит так:

- создается новая очередь в брокере
- настраивается маршрутизация
- в новой системе пишется обработчик загрузки
- источник продолжает отправлять те же сообщения

Это снижает связанность систем

Источник не содержит логику вида

- если получатель WMS — **отправить так**
- если получатель CRM — **отправить иначе**
- если получатель сайт — **отправить третьим способом**

Источник публикует событие или сообщение. **Получатели подключаются к этому потоку независимо**

Это одно из ключевых преимуществ брокерной архитектуры в RDV MI

Почему RDV MI не ESB?

RDV MI не является классической ESB и не пытается стать центральной шиной для всех систем предприятия

Его задача другая: дать 1С-команде стандартный интеграционный каркас для разработки, доставки, обработки, мониторинга и сопровождения обменов

Классическая ESB часто добавляет отдельный промежуточный слой преобразования

- источник
- формат шины
- преобразование
- формат приемника

RDV MI делает акцент на другом

- 1С формирует сообщение
- Брокер доставляет его получателям
- Приемник обрабатывает данные в своей прикладной логике
- Мониторинг показывает состояние обмена

Это снижает количество лишних преобразований и оставляет прикладную логику ближе к системам, которые реально знают свои данные

Где писать прикладную логику конвертации?

Прикладная логика пишется в конфигурации или расширении, в переопределяемых модулях и процедурах RDV MI

Базовая техническая часть уже есть

- Регистрация изменений
- Формирование сообщения
- Сериализация
- Отправка
- Получение
- Журналирование
- Мониторинг

Разработчик подключается там, где нужна прикладная специфика

- Изменить состав выгружаемых реквизитов
- Переопределить структуру JSON
- Добавить расчетные поля
- Описать поиск объекта при загрузке
- Выполнить проверки перед записью
- Обработать нестандартные ошибки
- Адаптировать данные под модель внешней системы

Такой подход сохраняет баланс: типовая интеграционная механика уже готова, но сложная бизнес-логика остается под контролем разработчика

Как RDV MI помогает ускорять обработку больших объемов?

RDV MI использует **очереди и отложенную обработку**

Это позволяет

- Обрабатывать входящие сообщения порционно
- Разделять разные типы сообщений по разным очередям
- Запускать независимые очереди параллельно
- Сохранять последовательную обработку там, где порядок критичен
- Не блокировать весь обмен из-за одного проблемного объекта

Важно: RDV MI не пытается автоматически распараллелить все подряд

В 1С параллельность требует аккуратности: блокировки, порядок зависимых объектов, ссылки и конкурирующая запись могут легко превратить ускорение в красивый пожар

Поэтому RDV MI дает инструменты, а **решение о параллельной обработке принимает разработчик или архитектор**

Как RDV MI проверяет, что данные действительно доехали корректно?

RDV MI поддерживает **контроль целостности данных**

Идея такая

- На стороне источника определяется набор реквизитов для контроля
- На стороне приемника определяется соответствующий набор данных
- По данным рассчитывается контрольное значение
- Результаты сравниваются
- Расхождения выводятся в отчет

Это позволяет проверить не только факт отправки сообщения, **но и конечное состояние данных**

RDV MI помогает ответить на вопросы

- Какие объекты отличаются между системами
- Какие объекты были выгружены
- Какие объекты не доехали
- Где есть расхождения по значениям
- Какие данные требуют повторной обработки или ручного анализа

Это важное отличие от обычного технического мониторинга: он показывает, что сообщение прошло маршрут, **а контроль целостности показывает, совпали ли данные по сути**

Как RDV MI связан с мониторингом?

RDV MI формирует сообщения и служебный контекст так, чтобы обмен можно было централизованно наблюдать

Для этого используется mi monitor — мониторинговый контур на базе **ClickHouse** и **Grafana**

Он показывает

- Сколько сообщений отправлено
- Сколько сообщений получено
- Какие базы онлайн или офлайн
- Где растут очереди
- Какие сообщения не доставлены
- Сколько времени занимает доставка
- Какие типы сообщений создают нагрузку
- Что находится внутри конкретного JSON-сообщения

Главная идея

RDV MI не просто отправляет данные. Он оставляет технический след, который можно анализировать

Это превращает сопровождение обменов из «посмотрите журнал регистрации» **в нормальную диагностику по дашбордам, статусам, очередям и конкретным сообщениям**

Можно ли использовать RDV MI только для простого HTTP-обмена без RabbitMQ?

Да. RabbitMQ не является обязательным для каждого сценария

Если есть простая интеграция между двумя системами, нет сложной маршрутизации, нет множества получателей и не требуется развивать событийный контур, HTTP может быть достаточным вариантом

Коротко:

- HTTP — для простых и точечных сценариев
- RabbitMQ — для развивающегося интеграционного контура

RDV MI не навязывает брокер там, где он не нужен. Но если обмены начинают расти, RabbitMQ дает больше гибкости и меньше связанности между системами

Насколько RDV MI открыт для доработок?

RDV MI построен **как расширяемый каркас**

Его можно адаптировать под проект:

- Добавлять прикладные модули
- Переопределять стандартные процедуры
- Менять формат сообщений
- Подключать новые сценарии загрузки и выгрузки
- Расширять транспортный слой
- Дорабатывать мониторинг
- Добавлять специфические проверки и контрольные механизмы

Ключевой принцип

RDV MI дает стандартную основу, но не забирает у команды контроль над кодом и архитектурой

Это не закрытая коробка, где любое отклонение требует обходных решений. **Это каркас, который закрывает типовую интеграционную механику и оставляет пространство для проектной специфики**