Nevel

ВНЕШНЕЕ ИСПОЛЬЗОВАНИЕ API

Ошибки API и повторные запросы

Как отличать ошибки доступа, лимитов, размера запроса и недоступности модели; какие запросы повторять.

Облако / В контуре

https://docs.nevel.sh/api/errors/

Обрабатывайте HTTP-статус вместе с телом ответа. Один статус может обозначать несколько причин: например, `429` возникает при ограничении частоты, исчерпании квоты или временном резервировании бюджета активными запросами. Пользовательское сообщение и стратегия повтора должны соответствовать конкретной причине и источнику отказа.

Формат ошибки

Ошибки шлюза возвращаются в объекте `error`. Он содержит описание `message`, тип `type` и, когда предусмотрено, машинный код `code`. Дополнительные поля могут указывать параметр запроса или технические подробности. Не стройте логику на точном английском тексте сообщения: используйте статус и код.

Ответ внешнего провайдера может иметь отличающуюся структуру. Клиент должен корректно обработать неизвестную ошибку и случай, когда промежуточный сетевой компонент возвращает текст вместо JSON. Сохраните статус и безопасное диагностическое описание, а пользователю покажите понятное следующее действие.

Основные случаи

HTTP	Код или ситуация	Действие
`400`	`invalid_request`	Проверить тело, обязательные `model` и `messages`
`401`	`invalid_api_key`	Проверить заголовок Bearer, состояние и срок ключа
`403`	`model_not_allowed`	Выбрать разрешённую модель или обратиться к администратору
`404`	`not_found`	Проверить метод, базовый URL и относительный путь
`413`	`payload_too_large`	Уменьшить размер тела запроса
`429`	`insufficient_quota`	Проверить бюджет и срок обновления квоты
`429`	`api_key_busy`	Учесть `Retry-After`, сократить параллелизм
`429`	Ограничение частоты, токенов или параллельности на шлюзе либо у провайдера	Определить область, уменьшить нагрузку и выполнить ограниченный повтор; код зависит от компонента

HTTP	Код или ситуация	Действие
502	`upstream_invalid_response`	Проверить временную ошибку ответа модели
504	`upstream_timeout`	Ограниченно повторить запрос с учётом его результата и стоимости
500	`server_error`	Сохранить диагностику; при повторении обратиться в поддержку

Таблица описывает основные случаи шлюза. Другие коды могут прийти от используемой модели или сетевой инфраструктуры. Если ошибка не соответствует ожидаемой, начните с минимального текстового запроса по быстрому старту, используя тот же адрес, ключ и модель.

Превышение размера

Ошибка `413 payload_too_large` от шлюза может содержать `error.metadata.body_limit_bytes`. Это ограничение размера HTTP-тела в байтах, а не длины контекста модели в токенах. Вложенное изображение в base64 увеличивает размер JSON; короткий текст запроса не означает, что всё тело небольшое.

Уменьшите вложение или разделите задачу на допустимые части. Если выбранная модель поддерживает получение изображения по URL, используйте только разрешённый политикой адрес с необходимыми правами. Адрес должен быть доступен именно обработчику выбранного маршрута. В on-prem внешний провайдер не получает доступ к внутреннему URL автоматически; разрешённый способ передачи файла проверяется отдельно. Для изолированного профиля обработчик и адрес остаются внутри периметра. Не делайте рабочий файл публичным ради обхода ограничения размера.

Когда повторять запрос

Ошибки авторизации, запрещённой модели и некорректного тела требуют изменения настроек или запроса. Немедленный повтор той же операции обычно не поможет. Для `insufficient_quota` проверьте лимиты, а для `api_key_busy` дождитесь освобождения зарезервированного бюджета и учитывайте `Retry-After`.

Для временных сетевых ошибок и отдельных `5xx` используйте ограниченное число попыток с увеличивающейся задержкой и случайным разбросом. Таймаут не доказывает, что модель не начала обработку. Повтор может привести к дополнительному расходу и другому ответу. Если приложение выполняет действия во внешних системах, защищайте эти действия от повторного исполнения отдельно.

Разбор временного ограничения

`429` от ограничителя частоты означает, что текущий запрос не допущен по соответствующему пределу. Это может быть RPM одного ключа, TPM пространства, занятый пул модели или ограничение внешнего провайдера. Сначала сократите нагрузку в этой области. Пауза одного потока не поможет, если остальные экземпляры приложения продолжают заполнять ту же очередь.

При неизвестном коде нельзя автоматически считать любой `429` временным: сначала исключите исчерпанный бюджет. Если доступна причина временного ограничения, используйте ограниченный повтор. При неопределённой причине сохраните диагностику и остановите бесконечный цикл. Не создавайте новый ключ и не меняйте провайдера для обхода правил; области rate limits описаны отдельно.

Ответ `429` может содержать `Retry-After`; обязательного наличия заголовка стандарт не устанавливает. Такой ответ нельзя сохранять для последующей выдачи как кэшированный результат API. RFC 6585, раздел 4.

Retry-After и случайная задержка

`Retry-After` бывает целым неотрицательным количеством секунд после получения ответа либо HTTP-датой. Клиент должен поддерживать обе формы. Например, `Retry-After: 12` означает ожидание 12 секунд; дата задаёт момент, до которого нужно ждать. Для даты учитывайте время UTC и рассинхронизацию часов. Невалидное значение не трактуется как разрешение непрерывно повторять запросы. RFC 9110, раздел 10.2.3.

Для временной ошибки без такого заголовка используйте exponential backoff с jitter: верхняя граница задержки растёт после неудачи, а конкретная пауза выбирается случайно внутри неё. Это распределяет повторы клиентов по времени. Одного одинакового удвоения паузы недостаточно: одновременно запущенные клиенты могут повторно встретиться в той же точке. Подход объяснён в AWS Architecture Blog.

Пример политики клиента, а не готовая конфигурация Nevel:

```
n — номер повторной попытки, начиная с 0
window = min(cap, base × 2^n)
если Retry-After корректен:
    delay = wait_from_header + random(0, window)
иначе:
    delay = random(0, window)
повтор разрешён только в пределах общего deadline и числа попыток
```

`base`, `cap`, deadline и предел попыток задаются владельцем интеграции по допустимому времени ответа. Ограничение `cap` в этом примере относится к случайной добавке, а не сокращает обязательное ожидание из заголовка. Если требуемая пауза выходит за deadline задачи, клиент завершает попытки и сообщает о временной недоступности. Истёкшая дата даёт нулевую нижнюю границу; случайная задержка и предел попыток сохраняются.

Повторы выполняются на одном выбранном уровне интеграции. Если SDK делает до трёх попыток на один вызов, а приложение вызывает его до трёх раз, итоговая нагрузка может стать девятикратной. При сетевой ошибке после отправки неизвестно, была ли операция принята. В таком случае перед повтором побочных действий проверяйте их состояние; произвольный заголовок идемпотентности не создаёт гарантию, если его поддержка не определена контрактом endpoint.

Повтор запроса не даёт разрешения менять получателя. В облаке и on-prem исключите скрытый fallback в клиенте, SDK и шлюзе. Заранее разрешённый резервный маршрут повторно проверяется для категории данных, а переключение фиксируется. При отсутствии допустимого резервного пути запрос завершается ошибкой; для изолированного профиля сбой локальной модели не открывает внешний маршрут. Варианты размещения и маршрутов описывают эти границы.

Потоковые ответы

При потоковой генерации успешные HTTP-заголовки могут прийти до завершения ответа. Последующий обрыв соединения нельзя считать полноценным успешным результатом. Сохраняйте признак завершения, различайте частичный и законченный ответ и сообщайте пользователю о прерывании. Автоматическое продолжение или повтор должно учитывать уже показанный текст и возможные действия приложения.

После начала ответа `200` статус этого же ответа не меняется на `429`. Ошибка приходит внутри поддерживаемого формата потока либо проявляется его прерыванием; формат зависит от маршрута. Не пытайтесь обнаруживать её только проверкой исходного HTTP-статуса. При повторе прерванного потока предыдущий текст не приклеивается к новой генерации как гарантированное продолжение: результат может отличаться, а первый запрос ещё обрабатываться у провайдера.

Проверка клиента

На тестовом ответе проверьте `429` с задержкой в секундах, HTTP-датой, отсутствующим и некорректным заголовком. Убедитесь, что при `insufficient_quota` повторов нет, временная перегрузка уменьшает подачу общей очереди, а достижение deadline прекращает ожидание. Для потока проверьте обрыв после нескольких фрагментов и отсутствие двойного выполнения связанного действия. Синтетические ответы тестового сервера проверяют логику клиента; они не подтверждают фактические лимиты или поведение конкретного провайдера.

Что передать в поддержку

Подготовьте время с часовым поясом, относительный путь API, модель, HTTP-статус, код ошибки и идентификатор запроса, если он возвращён. Добавьте частоту повторения и минимальное обезличенное тело, достаточное для воспроизведения. Не включайте ключ, заголовок авторизации или рабочие файлы целиком. Для on-prem укажите версию установки и изменения сети или конфигурации, после которых возникла ошибка.

Если речь об ограничении, приложите число активных запросов, частоту отправки, размер входа, источник `429` и полученный `Retry-After`, когда они известны. Отсутствующее поле отметьте как неизвестное; не восстанавливайте область лимита по одному статусу.