Nevel

ВНЕШНЕЕ ИСПОЛЬЗОВАНИЕ API

API-ключи и лимиты

Авторизация Bearer, области ограничений RPM и TPM, окна, всплески нагрузки, параллелизм и поведение потоковых запросов.

Облако / В контуре

https://docs.nevel.sh/api/auth-limits/

API-ключ предоставляет приложению программный доступ к моделям в рамках рабочего пространства. Он определяет контекст использования и позволяет отдельно управлять доступом и расходом интеграции. Ключ следует считать секретом: наличие пользовательского входа в платформу не заменяет его для программного запроса.

Авторизация запроса

Передавайте ключ в HTTP-заголовке `Authorization` с типом `Bearer`. Пример первого запроса читает его из `NEVEL_API_KEY`. Не передавайте ключ в URL: адрес запроса может попасть в историю, журналы промежуточных систем и диагностические материалы.

Ключ связан с конкретным пространством. Каталог моделей и ограничения проверяются для этого контекста. Если приложение получает отказ для модели, убедитесь, что используется ключ нужного пространства. Переключение пространства в браузере не изменяет переменную окружения запущенного приложения.

Ключ и маршрут к модели

API-ключ задаёт доступ приложения, но не определяет географию обработки. Место установки платформы и выбранный маршрут проверяются отдельно. В on-prem запрос приходит в платформу заказчика, после чего политика может разрешать локальную модель, российский API, self-hosted модель Nevel на серверах РФ или глобального провайдера. В облаке также доступны разрешённые для поставки self-hosted модели Nevel в РФ и провайдерские маршруты. Конкретный набор определяется каталогом и политикой пространства; матрица вариантов объясняет различия.

Собственный ключ провайдера меняет авторизацию и учёт, но не переносит его модель в инфраструктуру заказчика. Разрешение модели ключу не должно обходить запрет категории данных, список получателей или изолированный профиль. Основной и резервный пути проверяются одинаково: скрытое переключение на другой сервис при отказе не допускается.

Создание и назначение

Раздел API-ключей доступен при включённом управлении ключами. Администратор должен определить назначение ключа, владельца приложения, окружение и ожидаемую нагрузку. Используйте различимые названия, чтобы по списку ключей можно было понять, какому процессу принадлежит запрос.

Создавайте отдельные ключи для разработки и производственной работы. Для независимых приложений также удобнее иметь разные ключи: это позволяет отозвать один доступ без остановки остальных и отдельно ограничивать расход. Полученный секрет передаётся в хранилище секретов или переменные окружения серверного приложения.

Состояния и срок действия

Состояние	Значение для эксплуатации
`active`	Ключ активен; дополнительно проверяются срок действия, модель и ограничения
`disabled`	Ключ отключён; доступ следует считать приостановленным

Состояние	Значение для эксплуатации
`revoked`	Ключ отозван; для продолжения работы выдаётся новый доступ

У ключа может быть срок действия. Перед окончанием срока подготовьте замену, обновите секрет приложения и проверьте небольшой запрос. После подтверждения перевода отключите или отзовите старый ключ. При подозрении на раскрытие сначала прекращайте использование скомпрометированного доступа, затем проверяйте затронутые приложения и расход.

Лимиты расходов

Ключ может иметь собственный лимит с периодом `daily`, `weekly`, `monthly` или `lifetime`. Лимит и период задаются совместно. Период `lifetime` относится к суммарному использованию ключа без регулярного обновления бюджета. Для периодических лимитов ориентируйтесь на фактическое время следующего сброса, доступное в состоянии лимита, а не на предположение о локальной полуночи пользователя.

Лимит ключа не отменяет другие ограничения пространства и обслуживания. Даже при остатке бюджета конкретному ключу может быть запрещена модель. И наоборот, разрешённая модель не гарантирует доступность запроса после исчерпания бюджета. Уточняйте причину по HTTP-статусу и машинному коду ошибки.

Ограничения частоты и нагрузки

Rate limits ограничивают нагрузку за интервал или число одновременно выполняемых операций. Они действуют отдельно от денежного бюджета: короткие частые запросы могут исчерпать частоту, а один длинный запрос — доступную пропускную способность по токенам. Ниже описан механизм настройки и эксплуатации; приведённые примеры не задают квоты подписок или действующие численные лимиты Nevel.

Ограничение	Что измеряется	Что учитывает клиент
RPM / RPS	Количество запросов за минуту / секунду	Частоту запуска, включая повторные попытки
TPM	Токены в заданном минутном интервале	Размер входа, ожидаемый выход и правила учёта выбранного маршрута
Concurrency	Одновременно принятые незавершённые запросы	Длительность генерации, открытые потоки и освобождение слотов
Burst	Короткий разрешённый всплеск запуска	Мгновенную подачу очереди после простоя
Размер запроса	Байты HTTP-тела или контекст конкретной модели	Размер вложений и поддерживаемый контекст; это отдельная проверка

Счётчик запросов не равен счётчику токенов модели. Если на входе используется предварительная оценка токенов, профиль определяет её состав: вход и зарезервированный выход либо иной проверяемый способ. Для неизвестного выхода нельзя заранее считать фактическое конечное число. Итоговый учёт после завершения сверяется с доступными данными обработки; отсутствие финального usage при обрыве учитывается как незавершённый расчёт, а не как нулевое потребление.

Области действия

Ограничение связывается с конкретной областью: API-ключом, пространством или организацией, маршрутом и моделью, общим пулом обработки. Несколько ключей одного приложения могут расходовать общий ресурс пространства. Несколько моделей могут обслуживаться одним пулом. Поэтому доступность собственного ключа не гарантирует доступности общего пула.

Проверки применяются совместно: запрос должен удовлетворять каждому действующему ограничению. Число RPM одного ключа нельзя напрямую сравнивать с общим TPM и выбирать одно «минимальное» значение — это разные единицы и области. Полученный отказ должен разбираться в своей области; создание новых ключей не является способом обхода общего ограничения.

На внешнем маршруте дополнительно действуют ограничения получателя модели. Его `429` возможен после успешного допуска на стороне Nevel. Внутренние повторы шлюза также создают нагрузку получателю. Разделяйте входящие запросы приложения и исходящие попытки маршрута, чтобы не увеличивать нагрузку повтором одновременно в SDK, приложении и шлюзе.

Окна и короткие всплески

Фиксированное окно сбрасывает счётчик на установленной границе интервала. Из-за этого запросы в конце одного окна и начале следующего могут образовать заметный всплеск. Скользящее окно учитывает предшествующий интервал относительно текущего момента и ограничивает такой эффект. Формулировка «в минуту» без описания окна не позволяет точно определить время следующего допуска.

Token bucket использует накапливаемые разрешения: они поступают с заданной скоростью до установленной ёмкости, а запуск тратит разрешение. Ёмкость определяет burst; после простоя можно кратко запустить больше запросов, чем обычная скорость пополнения. Эти разрешения не являются текстовыми токенами LLM. Пример такого механизма описан в официальной документации Amazon API Gateway; параметры AWS не являются настройками Nevel.

Concurrency работает отдельно: даже при полном bucket новый запрос не принимается, если все слоты заняты. Burst ограничивает скорость старта, а concurrency — число незавершённых операций. Очередь клиента должна учитывать оба условия и иметь предел размера и времени ожидания.

Пример профиля для интеграции

Следующая таблица — **иллюстративная рекомендуемая конфигурация для испытательного стенда**, а не тариф, настройки по умолчанию или публичное обязательство Nevel. Её параметры позволяют воспроизвести разные виды отказа; производственные значения выбираются по фактическому маршруту и нагрузке.

Параметр примера	Значение	Смысл проверки
Частота одного ключа	60 запросов за скользящие 60 секунд	Отдельный предел количества запусков
Короткий всплеск ключа	Bucket на 5 разрешений, пополнение 1 в секунду	Ограничение резкого старта после простоя
Токены ключа	120 000 за скользящие 60 секунд	В этом примере резервируются вход и разрешённый максимальный выход
Параллельность ключа	4 незавершённых запроса	Пятый не запускается до освобождения слота
Завершение расчёта	Корректировка резерва по фактическому usage	Неиспользованный резерв освобождается по правилам профиля

Например, пять одновременно отправленных запросов не обязательно пройдут: у bucket достаточно разрешений, но предел параллельности допускает только четыре. Если один запрос резервирует 40 000 токенов, токеновый предел этого примера может остановить четвёртый раньше ограничения concurrency. Это независимые причины; увеличение только RPM их не устраняет.

Порядок допуска и завершения

1. Входной шлюз ограничивает чрезмерные тела и очевидную перегрузку до дорогостоящей обработки. Защита до авторизации может учитывать сетевой источник, но не заменяет лимиты приложения: за одним адресом могут находиться разные клиенты.
2. Сервер проверяет ключ, его состояние, пространство, разрешённую модель и маршрут. Неавторизованный запрос не получает модельные ресурсы.
3. Применяются действующие лимиты ключа, пространства и маршрута. Перед запуском резервируются необходимые токены и место в обработке; денежное резервирование остаётся отдельной проверкой.
4. После допуска выполняется вызов. Успех, ошибка и отмена завершают жизненный цикл: ресурсы освобождаются, фактическое использование учитывается, причина завершения сохраняется.

При нескольких экземплярах шлюза счётчики одной области должны оставаться общими, а допуск — не превышать ограничение из-за одновременного чтения старого остатка. В приёмке это проверяется запросами через разные экземпляры. Резервирование не оставляет занятый слот навсегда после падения процесса: эксплуатационный профиль определяет срок резерва, продление для живого запроса и восстановление зависшего состояния. Ошибка хранилища обязательных ограничений не должна незаметно превращать ограниченный маршрут в неограниченный.

Уже принятый запрос остаётся попыткой, даже если модель вернула ошибку: возврат временного резерва токенов не означает отмены факта запроса в RPM. Поведение отклонённых до допуска запросов фиксируется отдельно для входной защиты; клиент не должен рассчитывать, что бесконечные `429` бесплатны для всех счётчиков.

Потоки и длинные операции

Один потоковый запрос занимает один слот до завершения или фактического прекращения обработки. Каждый фрагмент SSE не является новым запросом и не обнуляет минутное окно. При разрыве клиентского соединения сервер освобождает слот после остановки либо завершения связанной работы; отправка отмены не доказывает мгновенную остановку внешнего провайдера.

Ограничения допуска проверяются до начала потока. После отправки HTTP-заголовков `200` сервер уже не может заменить статус того же ответа на `429`: последующая проблема обрабатывается как ошибка или прерывание потока по контракту маршрута. Клиент проверяет штатное завершение и не запускает повтор всей операции автоматически при каждом обрыве. Подробнее — ошибки и повторные запросы.

При `429` до начала обработки клиент разбирает причину и `Retry-After`, если заголовок присутствует. Сам HTTP-стандарт не определяет область счётчика и не требует этот заголовок в каждом ответе: RFC 6585, раздел 4. Имена дополнительных заголовков и кодов зависят от используемого ограничителя.

Параллельные запросы и резервирование

Код `api_key_busy` связан с **резервом денежного бюджета** и сам по себе не подтверждает, что достигнут RPM, TPM или предел concurrency из профиля частоты.

Запросы в обработке могут резервировать доступный бюджет. Поэтому остаток складывается с учётом уже потраченных и временно зарезервированных средств. При активных запросах новый вызов может получить `429` с кодом `api_key_busy`, даже если окончательные расходы ещё не достигли лимита.

В этом случае ограничьте параллелизм и учитывайте заголовок `Retry-After`, если он присутствует. После завершения текущих операций доступность бюджета пересчитывается. Не выдавайте автоматически новый ключ для обхода ограничения: сначала проверьте незавершённые запросы и согласованный бюджет приложения.

При `429 insufficient_quota` автоматический повтор не устраняет исчерпание квоты. Приложение должно остановить цикл повторов и сообщить владельцу о необходимости проверить бюджет или дождаться предусмотренного сброса периода.

Проверка перед запуском

Проверьте успешный запрос с действующим ключом, отказ с отключённым ключом, отказ для запрещённой модели и поведение при нескольких одновременных запросах. Убедитесь, что приложение не пишет секрет в журнал и различает временную занятость ключа и исчерпанную квоту. Для диагностики сохраняйте название приложения, время, статус и код ошибки; полный ключ в обращение поддержки не включается.

Для профиля частоты отдельно проверьте границу окна, всплеск после простоя, исчерпание TPM длинным запросом, занятые слоты, несколько ключей одного пространства и внешний `429`. Затем оборвите поток и убедитесь, что слот освобождается после завершения обработки; повторите проверку при падении обслуживающего процесса. Сравните ожидаемый и фактический допуск, область ограничения и задержки клиента. Численные результаты заносятся в протокол испытания, а не выводятся из примера профиля.

Численные лимиты конкретной установки задаются её конфигурацией; их нельзя определить по общему описанию HTTP или token bucket.