**Обработка ошибок**

Как gRPC справляется с ошибками и кодами ошибок gRPC.

Стандартная модель ошибки

Как вы могли заметить в нашем документе с описанием концепций и примерами, при успешном выполнении вызова gRPC сервер возвращает клиенту статус OK (в зависимости от языка статус OK может использоваться или не использоваться напрямую в вашем коде). Но что происходит, если вызов не удался?

В случае возникновения ошибки gRPC возвращает один из своих кодов состояния ошибки с дополнительным строковым сообщением об ошибке, в котором содержится дополнительная информация о произошедшем. Информация об ошибках доступна клиентам gRPC на всех поддерживаемых языках.

Более богатая модель ошибок

Описанная выше модель ошибок является официальной моделью ошибок gRPC, поддерживается всеми клиентскими и серверными библиотеками gRPC и не зависит от формата данных gRPC (будь то буферы протокола или что-то другое). Возможно, вы заметили, что она довольно ограничена и не позволяет передавать подробности об ошибках.

Однако если вы используете буферы протокола в качестве формата данных, вы можете рассмотреть возможность использования более подробной модели ошибок, разработанной и используемой Google, как описано здесь. Эта модель позволяет серверам возвращать, а клиентам использовать дополнительные сведения об ошибках, выраженные в виде одного или нескольких сообщений protobuf. Кроме того, она определяет стандартный набор типов сообщений об ошибках для удовлетворения наиболее распространённых потребностей (таких как недопустимые параметры, нарушения квот и трассировки стека). Двоичное кодирование этой дополнительной информации об ошибках в формате protobuf предоставляется в виде дополнительных метаданных в ответе.

Эта более подробная модель ошибок уже поддерживается в библиотеках C++, Go, Java, Python и Ruby, и, по крайней мере, в библиотеках grpc-web и Node.js есть открытые запросы на её поддержку. Другие языковые библиотеки могут добавить поддержку в будущем, если на неё будет спрос, так что, если вам интересно, проверьте их репозитории на GitHub. Однако обратите внимание, что библиотека grpc-core, написанная на C, скорее всего, никогда её не поддержит, поскольку она намеренно не зависит от формата данных.

Вы можете использовать аналогичный подход (добавлять сведения об ошибках в конечные метаданные ответа), если не используете буферы протокола, но вам, скорее всего, потребуется найти или разработать библиотеку для доступа к этим данным, чтобы использовать их в своих API.

Однако при принятии решения об использовании такой расширенной модели ошибок следует учитывать важные факторы, в том числе:

* Библиотечные реализации расширенной модели ошибок могут различаться в зависимости от языка с точки зрения требований и ожиданий в отношении полезной нагрузки с подробностями об ошибках
* Существующие прокси-серверы, регистраторы и другие стандартные обработчики HTTP-запросов не отображают сведения об ошибках и, следовательно, не могут использоваться для мониторинга или других целей
* Дополнительная информация об ошибках в трейлерах мешает блокировке начала очереди и снижает эффективность сжатия заголовков HTTP/2 из-за более частых промахов кэша
* Более объёмные сведения об ошибках могут не соответствовать ограничениям протокола (например, максимальному размеру заголовков), что приведёт к потере исходной ошибки

Коды состояния ошибки

Ошибки возникают в gRPC при различных обстоятельствах, от сбоев в сети до неподтверждённых подключений, и каждая из них связана с определённым кодом состояния. Во всех языках gRPC поддерживаются следующие коды состояния ошибок.

General errors

CaseStatus codeClient application cancelled the requestGRPC\_STATUS\_CANCELLEDDeadline expired before server returned statusGRPC\_STATUS\_DEADLINE\_EXCEEDEDMethod not found on serverGRPC\_STATUS\_UNIMPLEMENTEDЗавершение работы сервераGRPC\_STATUS\_UNAVAILABLEСервер выдал исключение (или сделал что-то другое, кроме возврата кода состояния для завершения RPC)GRPC\_STATUS\_UNKNOWN

Сбои в работе сети

КейсКод состоянияДанные не были переданы до истечения срока. Также относится к случаям, когда некоторые данные были переданы, но до истечения срока не было обнаружено других сбоевGRPC\_STATUS\_DEADLINE\_EXCEEDEDНекоторые данные, переданные (например, метаданные запроса были записаны в TCP-соединение) до разрыва соединенияGRPC\_STATUS\_UNAVAILABLE

Ошибки протокола

КейсКод состоянияНе удалось распаковать, но алгоритм сжатия поддерживаетсяGRPC\_STATUS\_INTERNALМеханизм сжатия, используемый клиентом, не поддерживается серверомGRPC\_STATUS\_UNIMPLEMENTEDДостигнуты предельные значения ресурсов управления потокомGRPC\_STATUS\_RESOURCE\_EXHAUSTEDНарушение протокола управления потокомGRPC\_STATUS\_INTERNALВозвращенный статус при анализе ошибкиGRPC\_STATUS\_UNKNOWNНе прошел проверку подлинности: учетным данным не удалось получить метаданныеGRPC\_STATUS\_UNAUTHENTICATEDНедопустимый хост, указанный в метаданных полномочийGRPC\_STATUS\_UNAUTHENTICATEDБуфер протокола ответа при разборе ошибокGRPC\_STATUS\_INTERNALОшибка синтаксического анализа буфера протокола запросаGRPC\_STATUS\_INTERNAL

Языковая Поддержка

Примеры кода на нескольких языках показывают, как работать со стандартными ошибками, а также с более подробными сведениями об ошибках.

ЯзыкПримерC++Пример обработки ошибок C ++Пример с подробными сведениями об ошибке C ++ВпередПример обработки ошибок GoПример с подробными сведениями об ошибке GoJavaПример обработки ошибок JavaПример с подробными сведениями об ошибке JavaУзелПример обработки ошибок узлаПитонПример с подробными сведениями об ошибке Python

Репозиторий grpc-errors также содержит дополнительные примеры обработки ошибок.

Последнее изменение: 29 февраля 2024 г.: руководство пользователя по кодам состояния (#1263) (fb661d4)