PR #52246 完整报告

vllm-project/vllm

[Bugfix][Anthropic] Return 4xx for client-caused errors in /v1/messages

合并时间：2026-08-16 16:05

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52246

执行摘要

- 一句话：Anthropic 入口将客户端错误统一映射为 4xx
- 推荐动作：值得精读。这是一个小而完整的『入口错误处理对齐共享框架』示例：评审推动作者放弃手写类型分支、复用 create_error_response，最终以极小 diff 完成语义修复，并配套了高质量回归测试（用真实 pydantic.ValidationError 复现 Issue 场景）。对维护其他 entrypoint 的团队，本 PR 展示了如何在共享异常框架演进时保持各入口一致，值得作为参考模式。

功能与动机

Issue #52088 报告：CI 的 Schemathesis 测试生成的 Anthropic 请求可包含多于 4 个 stop_sequences（Anthropic 侧 schema 允许），但 Anthropic→OpenAI 转换构造 ChatCompletionRequest 时抛出 pydantic ValidationError（OpenAI stop 字段最多 4 项）。api_router 的 except Exception 把该异常吞成 internal_error 导致 HTTP 500，与 Issue 中『Invalid user input should not result in HTTP 500』的预期不符。PR body 明确指出 schema 半部分已由 #51997 解决，本 PR 修复剩余的转换错误映射缺口。

实现拆解

1. 变更入口：vllm/entrypoints/anthropic/api_router.py。create_messages 与 count_tokens 两个路由的异常分支从「手写 JSON 500 + internal_error 响应体」改为 translate_error_response(create_error_response(e))。create_error_response 来自 vllm/entrypoints/serve/exception_handling/error_response.py（#52261 整合后的共享异常映射入口），按异常类型分类：pydantic.ValidationError（ValueError 子类）与 VLLMClientError 类客户端错误映射为 400 BadRequestError，其余异常保持 500。translate_error_response 负责把 OpenAI 风格 ErrorResponse 翻译成 Anthropic 错误格式（type + message），状态码沿用 ErrorResponse.error.code。同时删除对 sanitize_message 的直接 import——脱敏改由 create_error_response 内部完成。
2. 实现演进（4 个 commit）：首版（c90de82）手写 _translate_exception 类型分支区分客户端错误；评审（DarkLight1337）要求跟随 server_utils.py 的异常处理器逻辑；第二版（c21c8ed）删除手写类型检查，直接复用 create_error_response，净 diff 收敛为 +6/−26，并去掉不再使用的 VLLMClientError/ValidationError import；第三版（4bb117d）按 AGENTS.md 精简冗余注释；第四版（90b3250）rebase 到 #52261 后修正两处测试。

3. 测试配套：tests/entrypoints/anthropic/test_anthropic_messages_conversion.py。新增 TestClientErrorResponses 类，覆盖 4 个场景：转换期 ValidationError → 400 且错误 type 为 BadRequestError、错误消息保留『at most 4 items』；VLLMValidationError → 400；RuntimeError → 500（回归保护）；count_tokens 路由 → 400。测试用 _StubRequest（stop: Annotated[list[str], Field(max_length=4)] + 6 个元素）构造真实 pydantic.ValidationError，与线上 ChatCompletionRequest 校验行为等价。
4. 配套调整：tests/entrypoints/serve/exception_handling/test_error_sanitization.py。rebase 后 api_router 改为间接脱敏，从『必须直接调用 sanitize_message』的源级检查参数化清单中移除 vllm.entrypoints.anthropic.api_router。
5. 无配置 /schema/ 部署改动：路由的 OpenAPI responses 本就声明 400/500 两类响应，无需变更文档。

关键文件：

- vllm/entrypoints/anthropic/api_router.py（模块 请求路由；类别 source；类型 entrypoint；符号 create_messages, count_tokens, translate_error_response, attach_router）：PR 的核心改动文件：create_messages 与 count_tokens 两个路由的异常处理从手写 internal_error 500 改为复用共享 create_error_response 统一分类（客户端 4xx、服务端 500），并移除对 sanitize_message 的直接 import。
- tests/entrypoints/anthropic/test_anthropic_messages_conversion.py（模块 消息转换；类别 test；类型 test-coverage；符号 TestClientErrorResponses, _make_api_app, _request_body, _conversion_error）：新增 TestClientErrorResponses 回归测试类，覆盖转换期 ValidationError、VLLMValidationError → 400，RuntimeError → 500，以及 count_tokens 路由 → 400，是本 PR 语义修复的验证主体。
- tests/entrypoints/serve/exception_handling/test_error_sanitization.py（模块 错误脱敏；类别 test；类型 test-coverage；符号 TestAffectedModulesUseSanitize）：rebase 到 #52261 后 api_router 改由 create_error_response 间接脱敏，不再直接调用 sanitize_message，因此从源级检查的参数化清单中移除该模块。

关键符号：create_messages, count_tokens, translate_error_response, TestClientErrorResponses._conversion_error, TestClientErrorResponses.test_validation_error_returns_bad_request, TestClientErrorResponses.test_generic_error_still_returns_internal_server_error

关键源码片段

tests/entrypoints/anthropic/test_anthropic_messages_conversion.py

新增 TestClientErrorResponses 回归测试类，覆盖转换期 ValidationError、VLLMValidationError → 400，RuntimeError → 500，以及 count_tokens 路由 → 400，是本 PR 语义修复的验证主体。

```
# tests/entrypoints/anthropic/test_anthropic_messages_conversion.py
# 客户端导致的错误必须是 4xx，而不是 500（Issue #52088）
class TestClientErrorResponses:
    @staticmethod
    def _make_api_app(handler: MagicMock):
```

```
    app = FastAPI()
    attach_router(app)
    app.state.args = Namespace(log_error_stack=False)
    # 注册 FastAPI 请求校验异常处理器，保持 schema 校验错误 → 4xx 语义
    app.exception_handler(RequestValidationError)(validation_exception_handler)
    app.state.anthropic_serving_messages = handler
    return app

@staticmethod
def _request_body() -> dict:
    return {
        "model": "test-model",
        "max_tokens": 1,
        "messages": [{"role": "user", "content": "Hello"}],
    }

@staticmethod
def _conversion_error() -> ValidationError:
    """构造真实 pydantic ValidationError，等价于 ChatCompletionRequest
    构造时对 stop 字段超长的校验失败（复现 Issue #52088 场景）。"""

    class _StubRequest(BaseModel):
        stop: Annotated[list[str], Field(max_length=4)] | None = None

    with pytest.raises(ValidationError) as exc_info:
        _StubRequest(stop=["a"] * 6)
    return exc_info.value

def test_validation_error_returns_bad_request(self):
    # 转换阶段 pydantic ValidationError 应映射为 400，而非 500
    handler = MagicMock(spec=AnthropicServingMessages)
    handler.create_messages.side_effect = self._conversion_error()

    app = self._make_api_app(handler)
    with TestClient(app, raise_server_exceptions=False) as client:
        response = client.post("/v1/messages", json=self._request_body())

    assert response.status_code == HTTPStatus.BAD_REQUEST
    body = response.json()
    assert body["type"] == "error"
    assert body["error"]["type"] == "BadRequestError"
    assert "at most 4 items" in body["error"]["message"]

def test_vllm_client_error_returns_bad_request(self):
    # Serving 层抛出的 VLLMValidationError 也映射为 400
    handler = MagicMock(spec=AnthropicServingMessages)
    handler.create_messages.side_effect = VLLMValidationError(
        "Invalid value for stop", parameter="stop"
    )
```

```
        app = self._make_api_app(handler)
        with TestClient(app, raise_server_exceptions=False) as client:
            response = client.post("/v1/messages", json=self._request_body())

        assert response.status_code == HTTPStatus.BAD_REQUEST
        assert response.json()["error"]["type"] == "BadRequestError"

    def test_generic_error_still_returns_internal_server_error(self):
        # 非客户端错误必须保持 500，防止服务端故障被误报成客户端问题
        handler = MagicMock(spec=AnthropicServingMessages)
        handler.create_messages.side_effect = RuntimeError("boom")

        app = self._make_api_app(handler)
        with TestClient(app, raise_server_exceptions=False) as client:
            response = client.post("/v1/messages", json=self._request_body())

        assert response.status_code == HTTPStatus.INTERNAL_SERVER_ERROR
        assert response.json()["error"]["type"] == "InternalServerError"

    def test_count_tokens_validation_error_returns_bad_request(self):
        # count_tokens 路由同样需要 4xx 语义
        handler = MagicMock(spec=AnthropicServingMessages)
        handler.count_tokens.side_effect = self._conversion_error()

        app = self._make_api_app(handler)
        with TestClient(app, raise_server_exceptions=False) as client:
            response = client.post(
                "/v1/messages/count_tokens", json=self._request_body()
            )

        assert response.status_code == HTTPStatus.BAD_REQUEST
        assert response.json()["error"]["type"] == "BadRequestError"
```

评论区精华

核心讨论围绕『是否手写异常类型判断』展开：DarkLight1337 在本 PR 进行中提醒存在 #52261 异常处理重构，要求跟随 vllm/entrypoints/serve/utils/server_utils.py 的既有异常处理器逻辑，而不是在 api_router 里再维护一套类型检查。作者据此把首版手写的 _translate_exception 分支删除，改为直接调用 create_error_response(e)，让分类语义由共享模块统一保证；api_router 只负责把结果翻译成 Anthropic 格式。此外，rebase 到 #52261 后出现了两个行为变化需要同步：通用错误映射出的 type 从 internal_error 变为 OpenAI 风格 InternalServerError（测试断言随之更新）；sanitize_message 不再被 api_router 直接调用（源级检查清单随之移除）。流程上还有多次 merge 冲突提示与 /ci run、/ci retry 触发，最终由 DarkLight1337 Approve（『Thanks for fixing!』）。

- 复用共享异常处理 create_error_response 而非手写类型分支 (design): api_router 统一走 create_error_response 做异常分类，再经 translate_error_response 转成 Anthropic 错误格式；净 diff 收敛为 +6/−26。

- rebase 到 #52261 后的错误类型与 sanitize 检查变化 (design): 测试断言更新为 InternalServerError；test_error_sanitization.py 的参数化清单移除 api_router。
- merge 冲突与 CI 触发流程 (other): rebase 到最新 main（含 #52261）后无冲突合并，CI 通过。

风险与影响

- 风险：
  1. 错误类型字符串变化：rebase 到 #52261 后，通用异常映射出的错误 type 从原来的 internal_error 变为 OpenAI 风格的 InternalServerError，依赖该字符串做监控或客户端分支的用户会观察到差异；测试已同步更新，但线上兼容性仍值得留意。
  2. 依赖共享异常映射的语义：create_error_response 对异常的分类逻辑集中在 vllm/entrypoints/serve/exception_handling/error_response.py，其中 pydantic.ValidationError 因是 ValueError 子类被归为客户端错误。若 serving 层未来用 ValueError 表达服务端故障，会被误映射为 4xx；若共享模块调整分类规则，Anthropic 入口会自动继承新行为，属于「单一职责收益」与「隐式耦合风险」并存。
  3. 流式响应中途错误不在本映射范围：StreamingResponse 开始输出后发生的迭代异常发生在 try/except 之外，本次改动未覆盖，属既有行为，不是本 PR 引入的回归。
  4. 兼容性：路由 OpenAPI responses 原本就声明 400/500，无需文档或客户端适配；错误消息脱敏仍生效（间接调用 sanitize_message）。 - 影响：用户侧：Anthropic Messages API 与 count_tokens 的客户端错误从误导性的 500 变为准确的 4xx，便于客户端区分输入问题与服务端故障、制定重试策略；服务端 500 告警的信号价值提升。系统侧：消除了 api_router 中重复的手写错误构造代码，与 #52261 的整合架构对齐，后续异常分类改动只需在共享模块进行。团队侧：新增 TestClientErrorResponses 建立了 400/500 边界的回归防线，并同步维护了 test_error_sanitization.py 的源级检查清单，防止入口脱离统一脱敏路径。影响范围集中在 Anthropic 前端入口层，不涉及核心调度、注意力或模型代码。 - 风险标记：错误类型字符串变更，依赖共享异常映射框架语义，任意 ValueError 子类按客户端错误处理，流式中途错误不受本映射保护（既有行为）

关联脉络

- PR #51997 [Bugfix][Anthropic] Add max_length on stop_sequences（标题依 PR body 描述推断）：PR body 明确提到 #51997 解决了 Issue #52088 的 schema 半部分（给 stop_sequences 增加 max_length 约束），与本 PR 的转换错误映射修复共同闭环同一 issue。
- PR #52261 [Refactor] Consolidate entrypoint exception handling（标题依评审与 commit 描述推断）：评审 DarkLight1337 要求本 PR 对齐该异常处理整合重构；rebase 后复用其 create_error_response，并相应调整测试断言与 sanitize 源级检查清单。