PR #35225 完整报告

sgl-project/sglang

refactor: rename chat response token IDs

合并时间：2026-08-18 14:07

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35225

执行摘要

- 一句话：chat 响应 token_ids 改名 response_token_ids
- 推荐动作：值得精读。该 PR 展示了“最小化破坏性重命名”的标准套路：协议模型、响应构造、测试三点同步修改，并在 PR body 明确列出 Review Focus 和客户端迁移提示。值得关注的设计决策包括：不添加兼容别名、保持请求参数与内部字段不动、用断言锁死旧键不出现。后续可考虑是否需要在文档或变更日志中给出迁移指引。

功能与动机

PR body 明确指出：return_token_ids 产生的 chat completion 载荷存在语义错位，choices[*].token_ids 只包含生成出的响应 token IDs，而 prompt token IDs 已单独经 prompt_token_ids 返回；未限定的字段名容易被误认为包含全部 token IDs。作者刻意只重命名该 chat 响应键，保持 return_token_ids、内部 output_ids 与普通 completions 响应键不变，且不增加兼容别名或双键输出。

实现拆解

整体拆解为四步：

1. 协议模型重命名：python/sglang/srt/entrypoints/openai/protocol.py 中 ChatCompletionResponseChoice 的 token_ids 字段改为 response_token_ids，同时 _serialize 中 data.pop("token_ids", None) 改为 data.pop("response_token_ids", None)。这样当字段为 None 时新键不会出现在序列化结果中，旧键也永远不会再被输出。
2. 响应构造同步调整：python/sglang/srt/entrypoints/openai/serving_chat.py 的 OpenAIServingChat._build_chat_response 中，构造 ChatCompletionResponseChoice 时把 token_ids=choice_token_ids 改为 response_token_ids=choice_token_ids。choice_token_ids 的取值仍来自 ret_item["output_ids"] 且受 request.return_token_ids 控制，请求侧语义未变。
3. 测试三处联动更新：
   - test_protocol.py：序列化断言改为检查 response_token_ids，并新增 assertNotIn("token_ids", data) 确认旧键绝不会出现。
   - test_serving_chat.py：构造响应后同时断言对象属性与 model_dump() 结果中的新字段映射。

- test_sampling_mask.py：端到端采样掩码测试改为读取 choices[0]["response_token_ids"]。

4. 明确不变量：请求参数 return_token_ids、内部 output_ids、普通 /v1/completions 响应键均保持不变，且不提供兼容别名或双键输出，因此这是一次对外破坏性重命名，需客户端配合迁移。

关键文件：

- python/sglang/srt/entrypoints/openai/protocol.py（模块 协议层；类别 source；类型 core-logic；符号 ChatCompletionResponseChoice, _serialize）：对外响应协议的核心定义文件：ChatCompletionResponseChoice 的字段从 token_ids 改为 response_token_ids，并同步修改 _serialize 的按键弹出逻辑，决定序列化输出是否包含旧键。
- python/sglang/srt/entrypoints/openai/serving_chat.py（模块 chat 服务；类别 source；类型 core-logic；符号 OpenAIServingChat._build_chat_response）：响应构造的唯一入口：_build_chat_response 将内部 output_ids 映射到对外字段，本次把赋值目标从 token_ids 改为 response_token_ids，是运行时行为变化的实际发生点。
- test/registered/unit/entrypoints/openai/test_protocol.py（模块 协议测试；类别 test；类型 test-coverage；符号 test_prompt_token_ids_and_meta_info_serialization）：验证协议序列化层：既确认新键 response_token_ids 正常输出，又用 assertNotIn("token_ids", data) 锁死旧键不会残留。
- test/registered/unit/entrypoints/openai/test_serving_chat.py（模块 chat 测试；类别 test；类型 test-coverage；符号 test_non_streaming_chat_response_returns_requested_token_ids_and_meta_info）：验证服务层构造：确认 return_token_ids=True 时，output_ids 能正确映射到 response_token_ids，且 model_dump() 后序列化字段也保持一致。
- test/registered/sampling/test_sampling_mask.py（模块 采样测试；类别 test；类型 test-coverage；符号 test_chat_completions_returns_top_p_only_sampling_mask）：端到端回归测试：真实 HTTP 请求 /v1/chat/completions 后从响应 JSON 读取生成侧 token IDs，确保线上序列化路径使用新键。

关键符号：ChatCompletionResponseChoice._serialize, OpenAIServingChat._build_chat_response, test_prompt_token_ids_and_meta_info_serialization, test_non_streaming_chat_response_returns_requested_token_ids_and_meta_info, test_chat_completions_returns_top_p_only_sampling_mask

关键源码片段

python/sglang/srt/entrypoints/openai/protocol.py

对外响应协议的核心定义文件：ChatCompletionResponseChoice 的字段从 token_ids 改为 response_token_ids，并同步修改 _serialize 的按键弹出逻辑，决定序列化输出是否包含旧键。

```
class ChatCompletionResponseChoice(BaseModel):
    index: int
    message: ChatMessage
    # 省略 logprobs / finish_reason / matched_stop / hidden_states 等既有字段
    # prompt_token_ids 与 response_token_ids 分别承载输入侧与生成侧 token IDs
```

```
    prompt_token_ids: Optional[List[int]] = None
    response_token_ids: Optional[List[int]] = None # 由原 token_ids 改名而来
    meta_info: Optional[Dict[str, Any]] = None

    @model_serializer(mode="wrap")
    def _serialize(self, handler):
        data = handler(self)
        # None 字段在序列化时被弹出，避免输出空键
        if self.hidden_states is None:
            data.pop("hidden_states", None)
        if self.prompt_token_ids is None:
            data.pop("prompt_token_ids", None)
        # 新键为 None 时同样移除，旧键 token_ids 不再可能出现在输出中
        if self.response_token_ids is None:
            data.pop("response_token_ids", None)
        if self.meta_info is None:
            data.pop("meta_info", None)
        return data
```

python/sglang/srt/entrypoints/openai/serving_chat.py

响应构造的唯一入口：_build_chat_response 将内部 output_ids 映射到对外字段，本次把赋值目标从 token_ids 改为 response_token_ids，是运行时行为变化的实际发生点。

```
# 在 OpenAIServingChat._build_chat_response 中构造单个 choice
# choice_token_ids 仍来自内部 output_ids，受 return_token_ids 请求参数控制
choice_token_ids = (
    ret_item["output_ids"] if request.return_token_ids else None
)
choice_data = ChatCompletionResponseChoice(
    index=idx,
    message=ChatMessage(
        role="assistant",
        content=text if text else "",
        tool_calls=tool_calls,
        reasoning_content=reasoning_text if reasoning_text else None,
    ),
    logprobs=choice_logprobs,
    finish_reason=finish_reason["type"] if finish_reason else None,
    matched_stop=(
        finish_reason["matched"]
        if finish_reason and "matched" in finish_reason
        else None
    ),
    hidden_states=hidden_states,
    prompt_token_ids=choice_prompt_token_ids,
    response_token_ids=choice_token_ids, # 字段改名后语义与内容对齐
    meta_info=choice_meta_info,
)
```

test/registered/unit/entrypoints/openai/test_protocol.py

验证协议序列化层：既确认新键 response_token_ids 正常输出，又用 assertNotIn("token_ids", data) 锁死旧键不会残留。

```
  def test_prompt_token_ids_and_meta_info_serialization(self):
      """Test that prompt_token_ids and meta_info serialize only when set."""
      default_choice = ChatCompletionResponseChoice(
          index=0,
          message=ChatMessage(role="assistant", content="Hello"),
          finish_reason="stop",
      )
      default_data = default_choice.model_dump()
      self.assertNotIn("prompt_token_ids", default_data)
      # 未设置时新键 response_token_ids 同样不应出现
      self.assertNotIn("response_token_ids", default_data)
      self.assertNotIn("meta_info", default_data)

      choice = ChatCompletionResponseChoice(
          index=0,
          message=ChatMessage(role="assistant", content="Hello"),
          finish_reason="stop",
          prompt_token_ids=[1, 2, 3],
          response_token_ids=[4, 5],
          meta_info={"prompt_tokens": 3},
      )
      data = choice.model_dump()
      self.assertEqual(data["prompt_token_ids"], [1, 2, 3])
      # 旧键 token_ids 必须彻底消失
      self.assertNotIn("token_ids", data)
      self.assertEqual(data["response_token_ids"], [4, 5])
      self.assertEqual(data["meta_info"], {"prompt_tokens": 3})
```

评论区精华

评审记录中 review_comments_count 为 0，唯一一条评审是 hnyls2002 的 APPROVED（无 review body），因此没有实质的代码评论交锋。核心关注点集中在 PR body 的 Review Focus 三项自检：

- ChatCompletionResponseChoice._serialize 需保证省略 token IDs 时移除新键、旧键不可输出；
- OpenAIServingChat._build_chat_response 需保证 return_token_ids 仅把内部 output_ids 映射到 response_token_ids；
- 客户端契约提醒：读取 choices[*].token_ids 的调用方必须迁移到 choices[*].response_token_ids。

Issue 评论中 hnyls2002 发起了 /rerun-test test_sampling_mask.py test_protocol.py test_serving_chat.py，bot 回报 ubuntu-latest 上协议与 chat 服务测试全部通过，而 1-gpu-5090 上的 test_sampling_mask.py 失败；该失败与本次重命名逻辑无直接关联，PR

最终仍被批准合并。

- CI 重跑与采样掩码测试失败 (testing): 失败发生在 5090 单 GPU 环境，与本次字段重命名逻辑无直接关联，PR 最终被批准合并。

风险与影响

- 风险:
  - 对外兼容性风险（高）：/v1/chat/completions 非流式响应中 choices[*].token_ids 被移除且无 alias，任何依赖旧键的下游客户端、评估工具或监控脚本会静默取不到值（JSON 缺字段不报错），容易造成数据丢失。
  - 流式响应不受影响：流式 chunk 使用 DeltaMessage，本不含 token_ids 字段，因此重命名不影响流式路径。
  - 内部一致性风险（低）：请求参数 return_token_ids 与内部 output_ids 未改名，_build_chat_response 中映射关系保持一对一，无状态残留。
  - 测试盲区：测试覆盖了非流式协议序列化、serving_chat 构造和端到端 sampling_mask，但没有显式回归普通 /v1/completions 和流式响应，虽然理论上不受影响，但缺少断言锁。
  - 无性能与安全影响：纯字段重命名，不涉及计算路径。
- 影响:
  - 用户体验影响：调用 /v1/chat/completions 并开启 return_token_ids 的客户端需要改为读取 choices[*].response_token_ids；若文档未同步，迁移成本会被低估。
  - 系统影响：无运行时计算路径变化，REST 响应 JSON 的 schema 变化是唯一实质影响。
  - 团队 / 工程影响：这是小范围但公开契约的语义修正，属于 API 命名精确化方向的先例；后续可考虑引入兼容层或变更日志机制，避免无声破坏。
  - 风险标记：破坏性 API 变更，无兼容别名，客户端需迁移，测试覆盖非流式路径

关联脉络

- PR #35028 config: one control-plane log for the process: 同批次改动了 test_serving_chat.py 与 chat 服务相关测试，说明 chat 服务层近期在持续整理配置读取与响应契约语义，本 PR 与之属于同一演进方向。
- PR #35027 config: the readback and the resolving view say what they are: 同属 API 契约与命名语义精确化的系列改动，涉及 entrypoints 响应表面与 server_info 回读契约，与本次对响应字段语义的修正思路一致。