```
import logging

from a2a.server.agent_execution import AgentExecutor, RequestContext
from a2a.server.events import EventQueue
from a2a.server.tasks import TaskUpdater
from a2a.types import (InternalError, InvalidParamsError, Part, TaskState,
                       TextPart, UnsupportedOperationError)
from a2a.utils import new_agent_text_message, new_task
from a2a.utils.errors import ServerError
from .agent import CurrencyAgent

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)


class CurrencyAgentExecutor(AgentExecutor):
    """Currency Conversion AgentExecutor Example."""

    def __init__(self):
        self.agent = CurrencyAgent()

    async def execute(
        self,
        context: RequestContext,
        event_queue: EventQueue,
    ) -> None:
        error = self._validate_request(context)
        if error:
            raise ServerError(error=InvalidParamsError())

        query = context.get_user_input()
        task = context.current_task
        if not task:
            task = new_task(context.message)  # type: ignore
            await event_queue.enqueue_event(task)
        updater = TaskUpdater(event_queue, task.id, task.context_id)
        try:
            async for item in self.agent.stream(query, task.context_id):
                is_task_complete = item['is_task_complete']
                require_user_input = item['require_user_input']

                if not is_task_complete and not require_user_input:
                    await updater.update_status(
                        TaskState.working,
                        new_agent_text_message(
                            item['content'],
                            task.context_id,
                            task.id,
                        ),
                    )
                elif require_user_input:
                    await updater.update_status(
                        TaskState.input_required,
                        new_agent_text_message(
                            item['content'],
                            task.context_id,
                            task.id,
                        ),
                        final=True,
                    )
                    break
                else:
```

```
                await updater.add_artifact(
                    [Part(root=TextPart(text=item['content']))],
                    name='conversion_result',
                )
                await updater.complete()
                break

    except Exception as e:
        logger.error('An error occurred while streaming the response: '
                     f'{e}')
        raise ServerError(error=InternalError()) from e

def _validate_request(self, context: RequestContext) -> bool:
    return False

async def cancel(
    self, context: RequestContext, event_queue: EventQueue
) -> None:
    raise ServerError(error=UnsupportedOperationError())
```