AgentAdapterService

September 4, 2026

AgentAdapterService

Kind: Service

Source: `atloria-monorepo/apps/api/src/support-agent/agent-adapter.service.ts`

`AgentAdapterService` is a NestJS backend service that provides a unified interface for executing support-agent requests. It supports both complete responses through `execute()` and incremental output through `executeStream()`, allowing callers to choose between request-response and streaming workflows.

Methods

Method	Signature	Returns	Description
`execute`	`execute(message: string, executionConfig: AgentExecutionConfig)`	`Promise<AgentExecutionResult>`	Execute a single message with the agent (stateless)
`executeStream`	`executeStream(message: string, executionConfig: AgentExecutionConfig)`	`AsyncGenerator<string, void, unknown>`	Stream execution (returns async generator) Uses runtime.runStreamed() directly for proper streaming support

Dependencies

- `ConfigService`
- `PrismaService`

Where it refuses work

- `AgentAdapterService` stops the work with `Error` when `!this.runtime` — “Agent runtime not initialized - check Azure/Anthropic configuration”.

- `AgentAdapterService` stops the work with `Error` when `!this.runtime` — “Agent runtime not initialized”.
- `AgentAdapterService` stops the work with an early return when `!response.ok`.

When something fails

- `AgentAdapterService` handles failure in 1 place: it logs it and continues in all 1.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller as Support Agent Controller
  participant Service as AgentAdapterService
  participant Agent as Agent Provider

  Client->>Controller: Submit support request
  Controller->>Service: execute() or executeStream()

  alt Complete response
    Service->>Agent: Execute agent request
    Agent-->>Service: AgentExecutionResult
    Service-->>Controller: AgentExecutionResult
    Controller-->>Client: Complete response
  else Streaming response
    Service->>Agent: Start streaming request
    loop For each output chunk
      Agent-->>Service: Text chunk
      Service-->>Controller: yield chunk
      Controller-->>Client: Stream chunk
    end
  end
```

Usage

```
import { Injectable } from '@nestjs/common';
import { AgentAdapterService } from './agent-adapter.service';

@Injectable()
export class SupportAgentFacade {
  constructor(
    private readonly agentAdapterService: AgentAdapterService,
  ) {}

  async getResponse() {
    const result = await this.agentAdapterService.execute();
```

```
    return result;
  }

  async *streamResponse(): AsyncGenerator<string, void, unknown> {
    for await (const chunk of this.agentAdapterService.executeStream()) {
      yield chunk;
    }
  }
}
```

AI Coding Instructions

- Use `execute()` when the caller needs a complete `AgentExecutionResult` before responding.
- Use `executeStream()` with `for await...of` for chat, SSE, or other incremental-response integrations.
- Preserve the async contract: do not convert streaming output into a buffered response unless the consuming endpoint explicitly requires it.
- Keep provider-specific agent logic behind this adapter so controllers and other services depend on the adapter interface rather than agent implementation details.
- Ensure consumers handle errors and connection cleanup correctly when relaying streamed chunks to clients.

Relationships

- DEPENDS_ON → `configservice`
- DEPENDS_ON → `PrismaService`

Referenced By

- `SupportAgentModule` (MODULE_PROVIDES)
- `SupportAgentService` (DEPENDS_ON)