dotnet-bridge

September 4, 2026

dotnet-bridge

Kind: Module

Source: `dotnet-bridge`

.NET project in solution

`dotnet-bridge` is the .NET project within the solution responsible for exposing .NET-backed functionality to the rest of the system. It acts as an integration boundary, typically handling request/response serialization, .NET runtime execution, and communication with JavaScript or TypeScript consumers.

Diagram

```
graph TD
  A[TypeScript / JavaScript Application] --> B[dotnet-bridge]
  B --> C[Request Serialization]
  C --> D[.NET Bridge Handlers]
  D --> E[Domain Services / Libraries]
  E --> D
  D --> F[Response Serialization]
  F --> A
```

Usage

```
import { spawn } from "node:child_process";

const bridge = spawn(
  "dotnet",
  ["run", "--project", "./dotnet-bridge/dotnet-bridge.csproj"],
  { stdio: ["pipe", "pipe", "inherit"] },
);

bridge.stdout.on("data", (data) => {
  const response = JSON.parse(data.toString());
  console.log("Bridge response:", response);
```

```
});

bridge.stdin.write(
  JSON.stringify({
    action: "process",
    payload: { id: "example-id" },
  }) + "\n",
);
```

AI Coding Instructions

- Keep the bridge contract explicit: define stable request and response DTOs shared or mirrored across .NET and TypeScript boundaries.
- Serialize messages consistently, preferably as newline-delimited JSON when communicating through standard input/output.
- Avoid writing diagnostic logs to standard output if it is used for machine-readable responses; use standard error for logs and errors.
- Validate all incoming payloads at the bridge boundary before passing them to domain services.
- Keep .NET-specific implementation details inside `dotnet-bridge`; expose only transport-friendly data to JavaScript or TypeScript callers.

Referenced By

- `atloria` (MODULE_DECLARES)