Microsoft.CodeAnalysis.CSharp

September 4, 2026

Microsoft.CodeAnalysis.CSharp

Kind: Service

Source: `atloria-monorepo/libs/parser-core/src/dotnet-bridge/Atloria.CSharpParser/Atloria.CSharpParser.csproj`

NuGet package dependency

`Microsoft.CodeAnalysis.CSharp` is the Roslyn NuGet dependency used by the `Atloria.CSharpParser` .NET bridge to parse and analyze C# source code. It provides C# syntax trees, semantic analysis, diagnostics, and compiler APIs that the parser bridge exposes to the TypeScript-facing parser system.

Diagram

```
sequenceDiagram
    participant TS as TypeScript Parser Core
    participant Bridge as Atloria.CSharpParser (.NET)
    participant Roslyn as Microsoft.CodeAnalysis.CSharp
    participant Result as Parsed AST / Diagnostics

    TS->>Bridge: Submit C# source for parsing
    Bridge->>Roslyn: CSharpSyntaxTree.ParseText(source)
    Roslyn-->>Bridge: Syntax tree and compiler diagnostics
    Bridge->>Bridge: Convert Roslyn nodes to bridge DTOs
    Bridge-->>Result: Return AST, locations, and diagnostics
    Result-->>TS: Consume normalized parser output
```

Usage

```
import { execFile } from "node:child_process";
import { promisify } from "node:util";

const execFileAsync = promisify(execFile);
```

```ts
const parserProject =
  "libs/parser-core/src/dotnet-bridge/Atloria.CSharpParser/Atloria.CSharpParser.csproj";

async function parseCSharp(source: string) {
  const { stdout, stderr } = await execFileAsync(
    "dotnet",
    [
      "run",
      "--project",
      parserProject,
      "--",
      "--source",
      source,
    ],
    { maxBuffer: 10 * 1024 * 1024 },
  );

  if (stderr) {
    console.warn("C# parser diagnostics:", stderr);
  }

  return JSON.parse(stdout);
}

const result = await parseCSharp(`
public class Greeting
{
    public string Message => "Hello, Atloria!";
}
`);

console.log(result);
```

AI Coding Instructions

- Keep Roslyn-specific parsing and semantic-analysis logic inside the `Atloria.CSharpParser` .NET bridge; TypeScript consumers should use normalized bridge output rather than Roslyn types directly.
- Use `Microsoft.CodeAnalysis.CSharp` APIs such as `CSharpSyntaxTree.ParseText` and `CSharpCompilation` when syntax or semantic information is required.
- Preserve source spans, line/column locations, and Roslyn diagnostics when converting syntax nodes into transport DTOs.
- Avoid depending on unstable Roslyn internal node representations; serialize explicit DTO fields needed by the parser-core contract.
- Ensure the NuGet package version remains compatible with the target .NET SDK and other `Microsoft.CodeAnalysis.*` package references.

Referenced By

- `Atloria.CSharpParser` (DEPENDS_ON)