Pams.CurrencyConverter

August 18, 2026

Pams.CurrencyConverter

Kind: Module

Source: `Pams/Logic/Pams.CurrencyConverter/Pams.CurrencyConverter.csproj` (line 1)

Part of: Pams

.NET project in solution

`Pams.CurrencyConverter` is a .NET project in the Pams solution. It contains currency-conversion logic that other Pams projects can reference when they need to convert monetary values between currencies.

Diagram

```
graph TD
    PamsSolution[Pams Solution] --> CurrencyConverter[Pams.CurrencyConverter]
    Application[Consuming Pams Project] --> CurrencyConverter
    CurrencyConverter --> Conversion[Currency Conversion Result]
```

Usage

```
type CurrencyConverterClient = {
  convert(input: {
    amount: number;
    fromCurrency: string;
    toCurrency: string;
  }): Promise<number>;
};

async function convertInvoiceAmount(
  converter: CurrencyConverterClient,
  amount: number,
) {
  return converter.convert({
    amount,
```

```
    fromCurrency: "USD",
    toCurrency: "EUR",
  });
}

// Provide an adapter that calls the Pams.CurrencyConverter integration point.
const convertedAmount = await convertInvoiceAmount(currencyConverterClient, amount);
```

AI Coding Instructions

- Keep currency-conversion behavior within the `Pams.CurrencyConverter` project rather than duplicating conversion logic in consuming projects.
- Reference this project from Pams projects that require currency conversion.
- Define clear input and output contracts for amounts and currency codes at integration boundaries.
- Do not assume a conversion source, rate format, or external API contract without confirming the surrounding solution configuration.

Relationships

- DEPENDS_ON → `Newtonsoft.Json`
- DEPENDS_ON → `System`
- DEPENDS_ON → `System.Core`
- DEPENDS_ON → `System.Xml.Linq`
- DEPENDS_ON → `System.Data.DataSetExtensions`
- DEPENDS_ON → `Microsoft.CSharp`
- DEPENDS_ON → `System.Data`
- DEPENDS_ON → `System.Net.Http`
- DEPENDS_ON → `System.Xml`

Used by

3 references from 3 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Injected or called by (2)

- `Pams.API` — `Pams/API/Pams.API/Pams.API.csproj`:1
- `Pams.Business` — `Pams/Logic/Pams.Business/Pams.Business.csproj`:1

Declared in (1)

- `Pams` — `Pams/Pams.sln`:1