Porting Assistant for .NET

User Guide

Porting Assistant for .NET: User Guide

Table of Contents

What is Porting Assistant for .NET? 1
- Features of Porting Assistant for .NET 1
- Supported versions 2
- Concepts 2
- How to get started with Porting Assistant for .NET 2
- Accessing Porting Assistant for .NET 3
- Pricing for Porting Assistant for .NET 3

How it works 4
- Information collected 4

Get Started 6
- Prerequisites 6
  - Prerequisites 6
  - Memory requirements for the Porting Assistant for .NET Visual Studio for .NET IDE extension. 7
  - AWS Identity and Access Management (IAM) 7
- Install 10
- Use the assessment tool 10
  - Set up Porting Assistant for .NET 10
  - Assess a new solution 11
  - Analyze NuGet dependencies per project 12
- Port solution 12
- Remove solution assessment 13
- Change settings 13

Porting Assistant for .NET Visual Studio IDE extension 14
- Supported versions 14
- Prerequisites for using the Porting Assistant for .NET Visual Studio IDE extension 15
- Memory requirements 15
- Pricing for the Porting Assistant for .NET Visual Studio IDE extension 15
- How it works 15
- Install extension 17
- Assess and analyze solution 17
- Port solution 18
- Transition from standalone tool to extension 18
- Troubleshoot 19
- Version history 14

Security 20
- Data protection 20
  - Data collected 21
- Identity and Access Management 21
- Configuration and vulnerability analysis 22
- Security best practices 22

Collected Metrics 23

Version history 2

Document History 26

What is Porting Assistant for .NET?

Porting Assistant for .NET is a tool that helps you to port your existing .NET applications running on Windows Server to .NET Core on Linux. Porting Assistant for .NET scans .NET projects in an application portfolio, analyzes source code and package dependencies, and generates an assessment report that highlights incompatible APIs and packages (NuGet and Microsoft Core). Porting Assistant for .NET offers replacement suggestions for incompatible packages and APIs, where applicable. It also notifies you if the application is using unsupported components that cannot be easily replaced with .NET Core on Linux. The detailed compatibility assessment makes it possible for you to prioritize applications from your portfolio based on the complexity and effort involved in the porting process. When you select an application, Porting Assistant for .NET jumpstarts the porting process by converting the .NET project reference files to their .NET Core equivalent with the updated package information and versioning. Developers can use the updated project files as a starting point to make source code changes. The Porting Assistant for .NET suggestion engine for API and package replacements improves over time as it learns more about the usage patterns and frequency of missing packages and APIs.

The Porting Assistant for .NET Visual Studio IDE extension makes it possible to use Porting Assistant for .NET functionality seamlessly from within Visual Studio. For more information about the extensions, see Porting Assistant for .NET Visual Studio IDE extension (p. 14).

Contents

- Features of Porting Assistant for .NET (p. 1)
- Supported versions (p. 2)
- Concepts (p. 2)
- How to get started with Porting Assistant for .NET (p. 2)
- Accessing Porting Assistant for .NET (p. 3)
- Pricing for Porting Assistant for .NET (p. 3)

Features of Porting Assistant for .NET

Porting Assistant for .NET provides the following features:

Compatibility assessment

Porting Assistant for .NET scans your .NET framework projects and generates a compatibility assessment report by analyzing source code and packages (NuGet and Microsoft Core). Porting Assistant for .NET analyzes all third-party and internal packages and then classifies them into compatible and incompatible buckets. It identifies incompatible APIs and their source, and provides known replacements, if available. This assessment can help you to identify and prioritize appropriate applications for .NET Core porting.

Porting assistance

If the latest version of a package is compatible with .NET Core, Porting Assistant for .NET upgrades the package to its latest compatible version and changes relevant project reference files to a .NET Core-compatible format. Porting Assistant for .NET doesn't eliminate the need to make source code changes. However, it reduces the undifferentiated heavy-lifting required to begin refactoring source code.

Continuous improvement

The Porting Assistant for .NET API replacement engine, powered by the AWS Cloud, continuously learns and improves as the tool is used. The learning capability of the tool results in increasingly better ways to port applications to .NET Core.

Supported versions

Porting Assistant for .NET supports assessment and conversion of C# applications running on Windows Server 2016 and later.

- Supported .NET Framework source versions: .NET Framework 3.5 and later

  Note
  Within the source projects, only .csproj files are supported. MS Build conditional constructs and other conditional or nested clauses in .csproj files are not supported. Only NuGet version strings that conform to SemVer are supported. Version strings that use wildcards or variable names are not supported.
- Supported .NET Core target platform versions:
  - .NET Core 3.1 on Linux
  - .NET 5.0 on Linux
- Supported Operating Systems: Windows 10 or Windows Server 2016

Concepts

The following terminology and concepts are central to your understanding and use of Porting Assistant for .NET.

Solution/solution file

A solution or solution file is a collection of projects (`csproj`) packaged by Visual Studio with the extension `.sln`. Using a solution file makes it easier to assess large projects with many projects as opposed to analyzing individual binaries and then aggregating information.

`csproj`/project

Each project is identified by Visual Studio with a `.csproj` file. All of the project configuration is contained in the `.xml` file.

Internal NuGet

Some users have an internal NuGet hosting service, similar to nuget.org, that is accessible only within their local network.

How to get started with Porting Assistant for .NET

After you have downloaded (p. 10) and configured Porting Assistant for .NET, you can start the compatibility assessment in the **Application-level assessment** mode. You can select a local source-code folder and associated solution (.sln) file to start an assessment for this particular .NET framework application.

After starting the assessment job, it may take a few minutes to generate the assessment report. When the job is complete, you can select **View Details** to view the full compatibility report for the application. This compatibility report is the starting point for your .NET refactoring efforts and enables you to estimate the complexity of your application and the effort involved in porting it to .NET Core.

Accessing Porting Assistant for .NET

Porting Assistant for .NET is offered as a standalone tool that you download and install on your developer workstation or as a Visual Studio IDE extension. Using the standalone tool, you can specify the solution files for your .NET applications to start an assessment task. You can view and analyze the assessment report using a UI console and can optionally download it for sharing and offline analysis. When you use the IDE extension, you can perform assessments and porting from within your IDE, with guidance as you make source code changes.

Pricing for Porting Assistant for .NET

Porting Assistant for .NET is available for use at no cost.

How Porting Assistant for .NET works

Porting Assistant for .NET analyzes source code for supported APIs and packages (Microsoft Core APIs and packages, and public and private NuGet packages). It identifies incompatible API calls made from each package and checks whether a package is compatible with .NET Core. Porting Assistant for .NET parses package reference files for each project in the .NET Framework solution, and iterates through each referenced public and private NuGet package, as well as each Microsoft Core API and package, to check whether the latest version of the package that is compatible with .NET Core is available. To speed up the process, Porting Assistant for .NET includes a precompiled database of all public NuGet packages and Microsoft Core APIs and packages and their compatibility status. For private packages, Porting Assistant for .NET uses dotnet tools to determine the compatibility of a package. After compatibility for all packages is determined, you can view the package status at both the project and solution level, as a graph that shows package dependencies.

Porting Assistant for .NET uses the compatibility information of the NuGet packages and Microsoft Core packages and APIs to assign a compatibility score, which is an estimation of the effort required by the application to port the application to .NET Core. You can use the compatibility score to identify and prioritize applications for your porting project.

When you select an application for porting, Porting Assistant for .NET can speed up the process by setting up the project file with updated NuGet/Microsoft packages and formatted package references in a format that is compatible with .NET Core. You can use the updated project file to improve your application porting time. When you select an application for porting, Porting Assistant for .NET copies the .NET Framework source code and the associated project files to a .NET Core compatible format. If there are any known replacements, Porting Assistant for .NET applies them. When a project is ported, it might not be entirely .NET Core compatible because there may be other APIs, packages, and code blocks that must be substituted and refactored for compatibility.

Information collected

When you scan your source code, Porting Assistant for .NET collects the compatibility information on the public NuGet packages and the APIs within the packages that are in use by your application. Porting Assistant for .NET doesn't collect source code or information about your private NuGet packages. In the case of failure, the tool might collect stack traces to improve product experience. You have the option to disable telemetry gathering from within the tool at any time in the **Settings** menu of the Porting Assistant for .NET tool.

If you accept the data collection option in the **Settings** menu of the Porting Assistant for .NET tool, the following application data is collected:

1. Application errors generated when running assessments, porting, or when performing other functions provided by the Porting Assistant for .NET tool.
2. Names and versions of public NuGet packages assessed by the Porting Assistant for .NET tool.
3. Metrics for assessments run by the Porting Assistant for .NET tool on public NuGet packages, such as the number of packages and solutions, and the amount of time taken to create a solution.

Porting Assistant for .NET leverages the information on the public NuGet packages and APIs to continuously improve its API replacement suggestions. Porting Assistant for .NET periodically analyzes

the collected information and updates its replacement engine so that the experience continuously improves.

Getting started with Porting Assistant for .NET

This section contains information to help you set up your Porting Assistant for .NET environment, and to start assessing, converting, and porting your .NET Framework application to .NET Core.

Section topics

- Prerequisites (p. 6)
- Install Porting Assistant for .NET (p. 10)
- Use the Porting Assistant for .NET assessment tool (p. 10)
- Port a solution (p. 12)
- Remove a solution assessment from Porting Assistant for .NET (p. 13)
- Change your Porting Assistant for .NET settings (p. 13)

Prerequisites

The following prerequisites must be verified in order to successfully port your existing .NET applications to .NET Core using the Porting Assistant for .NET tool.

Topics

- Prerequisites (p. 6)
- Memory requirements for the Porting Assistant for .NET Visual Studio for .NET IDE extension. (p. 7)
- AWS Identity and Access Management (IAM) (p. 7)

Prerequisites

The Porting Assistant for .NET tool requires the following prerequisites for installation and usage.

- **.NET Core SDK 3.1**: Download .NET Core.
- **AWS CLI**: You must have a valid AWS CLI profile in order for Porting Assistant for .NET to collect compatibility information on the public NuGet packages and the APIs within the packages that are in use by your application. To view the type of application data collected by Porting Assistant for .NET , see Data collected by Porting Assistant for .NET (p. 21). Information about public NuGet packages is collected to help AWS prioritize work to address .NET Core incompatibilities on the NuGet packages, if any. For instructions on how to configure an AWS CLI profile, see AWS Identity and Access Management (IAM) (p. 7) .
- Windows 7 or later
- Internet connectivity on the machine running the assessment
- If your machine is behind a proxy with restricted internet access, you must permit access to the following:
  - **Amazon S3 Datastore** — `https://s3.us-west-2.amazonaws.com/aws.portingassistant.dotnet.datastore/`

- **Amazon S3 Datastore authenticaiton endpoint** — `https://encore-telemetry.us-east-1.amazonaws.com/`
- **GitHub datastore** — `https://github.com/aws/porting-assistant-dotnet-datastore/data`
- **GitHub user content** — `http://raw.githubusercontent.com/`
- **NuGet or additional package addresses** — any addresses required to restore your project packages (the most common one being `api.nuget.com`)

- Administrator access
- Processor with 1.8 GHz or above processing speed
- 4 GB minimum of available memory
- 5 GB minimum of free disk space

Memory requirements for the Porting Assistant for .NET Visual Studio for .NET IDE extension.

The following memory requirements must be met to use the Porting Assistant for .NET Visual Studio for .NET IDE extension.

Solution size	Minimum memory requirements
Small solutions (1,000 to 50,000 lines of code)	4 GB
Medium solutions (50,000 to 400,000 lines of code)	8 GB
Large solutions (400,000 or more lines of code)	16 GB or more, depending on size of source code

Note
These requirements are provided as estimates. Individual solutions can vary for memory usage.

AWS Identity and Access Management (IAM)

You must create an IAM user and attach a policy to your IAM user in order to grant permissions to the Porting Assistant for .NET service. Follow the steps in this section to create the user and attach the policy.

Topics

- Create the IAM user (p. 7)
- Attach required policy to the IAM user (p. 8)
- Create access keys for your IAM user (p. 9)
- Configure your AWS profile (p. 9)

Create the IAM user

1. Sign in to the AWS Management Console and navigate to the IAM console.
2. Choose **Users** from the left navigation pane.
3. Choose **Add user**.

4. Enter the user name for the new user. This is the sign-in name for AWS.
5. Under **Select AWS access type**, select **Programmatic access**.
6. Choose **Next: Permissions**.
7. Choose **Next: Tags**.
8. Choose **Next: Review** to view your configuration. After you have verified your selections, choose **Create user**.
9. Save the user access key IDs and secret access keys. You will use these to create AWS profiles. To view the user access keys (access key and secret access keys), choose **Show** next to each password and access key that you want to see. To save the access keys, choose **Download.csv** and save the file to a secure location.

   Important
   You will not have access to the secret keys again after this step.

For more information about creating IAM users, see Creating an IAM user in your AWS account.

Attach required policy to the IAM user

To grant the required permissions to use Porting Assistant for .NET, you must attach the following IAM policy to your IAM user.

1. Sign in to the AWS Management Console and navigate to the IAM console.
2. Choose **Policies** from the left navigation pane.
3. Choose **Create policy**.
4. Choose the **JSON** tab and copy and paste the following policy into the text field.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EnCorePermission",
            "Effect": "Allow",
            "Action": [
                "execute-api:invoke",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:492443789615:3dmmp07yx6/*",
                "arn:aws:execute-api:us-east-1:547614552430:8q2itpfg51/*",
                "arn:aws:s3:::aws.portingassistant.dotnet.datastore",
                "arn:aws:s3:::aws.portingassistant.dotnet.datastore/*"
            ]
        }
    ]
}
```

5. Choose **Next: Tags**.
6. Choose **Review Policy** and enter a **Name** and **Description** for the policy.
7. Choose **Create Policy**.
8. Filter the list of policies with the name of the policy you just created.
9. Select the bullet next to your new policy, and from the **Policy actions** dropdown, select **Attach**.
10. Select the **User name** of the IAM user created in Create the IAM user (p. 7).
11. Choose **Attach policy**.

Create access keys for your IAM user

If you are unable to locate the access keys for the IAM user to which the required policy is attached, perform the following steps.

1. Sign in to the AWS Management Console and navigate to the IAM console.
2. Choose **Users** from the left navigation pane.
3. Choose the name of the user to which the policy is attached, and then choose the **Security credentials** tab.
4. In the **Access keys** section, chose **Create access key**.

   Note
   If you already have an access key but cannot access your secret access key, make the old key inactive and create a new one.
5. To view the new access key, choose **Show**. You will not have access to the secret access key again after this dialogue box is closed. Your credentials will look something like the following example ID and access key:

   ```
   Access key ID: AKIAIOSFODNN7EXAMPLE

   Secret access key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   ```
6. To download the key pair, choose **Download .csv file**. Store the keys in a secure location. You will not have access to the secret access key again after this dialogue box is closed.

   Keep the keys confidential to protect your AWS account and never email them. Do not share them outside of your organizaiton, even if an inquiry appars to come from AWS or www.amazon.com. No one who legitimatley represents Amazon will ever ask you for your secret key.

For more information about creating an access key ID and secret access key, see Access key ID and secret access key.

Configure your AWS profile

After you have created an IAM user, you can configure your AWS named profile to apply settings and credentials to be applied when you run commands.

Configure AWS profile in the assessment tool

1. In the Porting Assistant for .NET assessment tool, navigate to **Set up Porting Assistant for .NET**.
2. Choose **Add a profile** under **AWS named profile**.
3. Enter your new **Profile name**, **AWS access key ID**, and **AWS secret access key**.
4. Choose **Add**.

Configure AWS profile using the AWS CLI

1. Run the following AWS CLI command to create a profile for Porting Assistant for .NET. The profile is named `default` in the credentials file.

   ```
   aws configure
   ```
2. For each prompt, enter the corresponding information.

   - `AWS Access Key ID`
   - `AWS Secret Access Key`

- `Default region name` (For example, `us-west-2`)
- `Default output format`

3. After you configure the profile using the AWS CLI, Porting Assistant for .NET will display the `default` profile under **AWS named profile** on the **Set up Porting Assistant for .NET** page of the assesment tool.

For more information about configuring the AWS CLI, see Configuring the AWS CLI .

Install Porting Assistant for .NET

Porting Assistant for .NET is available for download as an executable file. You can download Porting Assistant for .NET from the following location:

https://s3.us-west-2.amazonaws.com/aws.portingassistant.dotnet.download/latest/windows/Porting-Assistant-Dotnet.exe

After you have downloaded the installer, run the installation executable on your local computer to install Porting Assistant for .NET. When the installation completes, the Porting Assistant for .NET assessment tool automatically starts. For subsequent starts, open the **Start** menu and search for the Porting Assistant for .NET folder. The Porting Assistant for .NET tools appear under this folder.

Use the Porting Assistant for .NET assessment tool

This section contains information to help you get started with the Porting Assistant for .NET assessment tool. When you start the assessment tool for the first time, you are prompted to enter your AWS CLI profile information so that the Porting Assistant for .NET can collect metrics to improve your experience. These metrics also help flag issues with the software for AWS to quickly address. If you have not set up your AWS profile, see Configuring the AWS CLI.

Assessment tool topics

- Set up Porting Assistant for .NET (p. 10)
- Assess a new solution (p. 11)
- Analyze NuGet dependencies per project (p. 12)

Set up Porting Assistant for .NET

The following steps guide you through the Porting Assistant for .NET settings for collecting solutions data.

1. Verify the Prerequisites (p. 6).
2. From the **Set up Porting Assistant for .NET** page of the assessment tool, select an **AWS name profile** from the dropdown list or **Add a named profile**. Porting Assistant for .NET uses your AWS profile to share your solution information with AWS to make the Porting Assistant for .NET tool better.
3. Under **Porting Assistant for .NET data usage sharing**, clear the check box if you do not want to share your Porting Assistant for .NET solution information with AWS.

   If you accept the data collection option, the following application data is collected:

   - Application errors generated when running assessments, porting, or when performing other functions provided by the Porting Assistant for .NET tool.

- Names and versions of public NuGet packages assessed by the Porting Assistant for .NET tool.
- Metrics for assessments run by the Porting Assistant for .NET tool on public NuGet packages, such as the number of packages and solutions, and the amount of time taken to create a solution.

You can change your data collection settings at any time in the **Settings** menu.

4. Choose **Next** to assess your solution.

Assess a new solution

The following steps guide you through the creation of a new assessment solution.

1. From the **Assess a new solution** page of the assessment tool, **Specify a solution file path**. Choose your source solution (.sln) file to start an assessment for your .NET Framework application.

   Note
   The project is assessed using the version selected in your settings. The default version is .NET Core 3.1.

2. Choose **Assess**. You will be taken to the **Assessed solutions** page, where you can view a list of your solutions and the following details about each solution:

   - **Ported projects** — The number of projects in the solution that have been ported.
   - **Incompatible packages** — The number of packages in the solution that are incompatible with .NET Core .
   - **Incompatible APIs** — The number of API calls in the solution that are incompatible with .NET Core.
   - **Build errors** — The number of build errors in a solution after it has been assessed.

3. The status of your assessment appears at the top of the page. Choose **View details** to see an overview of your solution assessment, which includes a breakdown of the incompatible NuGet packages and APIs. You can also view the project references and source files from this page.
4. From the solution assessment page, choose **Export assessment report** to download a .csv file with your report details. Under the overview, you can choose the following tabs:

   - **Projects** — this tab lists the projects in your solution, the **Target framework** for the solution, the number of **Referenced projects** in the solution, the number of **Incompatible packages** in the solution, the number of **Incompatible APIs** in the solution, the **Portability score** (the number of compatible APIs over the number of incompatible APIs in the solution, represented as a percentage), and the **Port status** (ported or not ported) of the solution.
   - **Project references** — this tab displays the project references graph, which is a graphical representation of your projects and references. Select a node to see its project dependencies. To port your projects, we recommend that you start by selecting the base of your library, and then moving outward to test each layer. You should first port base libraries with the most dependencies from other projects. In other words, port libraries that show more inward arrows than outward arrows.

     To view details about a node, select the node and choose **View details**.
   - **NuGet packages** — this tab lists the number of NuGet packages in your solutions file, the **Name** of the NuGet packages, the **Version** number of the packages, the **Source files** in the package, the number of compatible **APIs** in the package, the compatibility **Status** of each package (compatible/incompatible), and the **Suggested replacement** for each package.
   - **APIs** — this tab lists the **Name** of each API call in the solution, the name of the **Package** within which the API call appears, the number of **Source files** that include each API call, the **Suggested replacement** for the API call, and the compatibility **Status** of the API call (compatible/incompatible).

Note
A small number of APIs might show an "Incompatible" status. This can happen when an API is not found in the Porting Assistant for .NET database and the status is unknown to the Porting Assistant for .NET system.

- **Source files** — this tab lists the **Source file name** in the solution, the number of **Incompatible API calls** over the total number of API calls for each source file, and the **Portability score** of the source file, which is the number of compatible APIs over the number of incompatible APIs in the solution, represented as a percentage. You can select a source file to view the incompatible API calls and replacement suggestions in the source code.

  In each source file, any section of code that is detected as incompatible will be highlighted as follows:

  - **Porting action** — code that initiates a porting action in the project.
  - **Incompatible method invocation** — an API that is incompatible with .NET Core.

5. After you make changes to your solution file, you can choose **Reassess solution** to start a new assessment of your solution.

Analyze NuGet dependencies per project

The following steps guide you through an analysis of the NuGet dependencies per project within an assessment.

1. From the main page of Porting Assistant for .NET, select **Assessed solutions** from the left navigation pane.
2. On the **Assessed Solutions** page, select a solutions file. You will be directed to the**Assessment overview** page.
3. Choose a project from the list at the bottom of the page. You will be directed to the **NuGet dependencies** page.
4. The NuGet dependencies page displays a list of compatible and incompatible NuGet packages, as well as each NuGet package **Version**, the number of **Projects** that include each NuGet package, the number of **Source files** that include each NuGet package, the number of compatible **APIs** in the package, the compatibility **Status** of each package (compatible/incompatible), and the **Suggested replacement** for each package. Porting Assistant for .NET identifies potential .NET Core replacement libraries with similar or identical APIs for some incompatible dependencies.
5. To view a graphical representation of the information, choose the **Project references** tab. This tab shows a graphical representation of your projects and references. Select a node to see its project dependencies. To port your projects, we recommend that you start by selecting the base of your library, and then moving outward to test each layer. You should first port base libraries with the most dependencies from other projects. In other words, port libraries that show more inward arrows than outward arrows.

Port a solution

To port a solution, follow these steps:

1. From the main page of Porting Assistant for .NET, choose **Get started**.
2. On the **Edit settings** page, choose the target .NET framework and AWS named profile to allow Porting Assistant to assess your solution. You can also add the AWS named profile using the AWS CLI.
3. From the main page of the assessment tool, select **Assessed solutions** from the left navigation pane.

4. On the **Assessed Solutions** page, select a solutions file. You will be directed to the**Assessment overview** page.
5. Under the **Projects** tab, select the project you want to port. Choose **Port project**. The Porting Assistant will ask how you want to save your ported project. Select whether you want to copy the ported project to a new location or modify the project in place.

   Note
   The project is ported to the version used in the assessment. If you want to port to a different version, you must reassess the project after changing the version in your settings.

   After you select the destination folder and choose **Save**, or select to modify the project file in place, you will be directed to the **Port projects** page.
6. Porting Assistant begins to port the new solution, and you are directed to the **Assessment overview** page. The status of the port appears at the top of the page. You can view the port status of a package by selecting it on the **Assessment overview** page and looking at the **Port status** in the overview section.

   Important
   When you port a solution, your project files and code are modified. Your project file is modified to include compatible packages you selected and other packages detected by the assessment. In addition, Porting Assistant adds or backs up code files based on the type of projects detected. Some code files are changed to make them more compatible with .NET Core. The result is not a completely ported project. The project may not build, and additional source code changes may be required. Any added or modified code must be verified and tested before it can be considered production ready.

Remove a solution assessment from Porting Assistant for .NET

To remove a solution assessment, follow these steps.

1. Navigate to the **Assessed solutions** page from the left navigation pane.
2. Select the bullet next to the solution assessment that you want to remove, and from the **Actions** dropdown, select **Remove**. When you remove a solution assessment, only the assessment report is removed from Porting Assistant for .NET. Your source code is not deleted.

Change your Porting Assistant for .NET settings

To change your user settings, perform the following steps.

1. Select **Settings** from the left navigation pane of the Porting Assistant for .NET tool.
2. On the **Settings** page, select **Edit**.
3. On the **Set up Porting Assistant for .NET** page, you can update your AWS user profile, your usage data selection, and the .NET Core version to use for the assessment.

Porting Assistant for .NET Visual Studio IDE extension

The Porting Assistant for .NET Visual Studio IDE extension makes it possible to use Porting Assistant for .NET functionality seamlessly from within Visual Studio. Once installed, the Porting Assistant for .NET extension provides prescriptive guidance to assist you with assessing and porting your Windows .NET Framework applications to .NET Core on Linux. This extension facilitates collaboration with other developers who are analyzing, debugging, testing, and refactoring the same application code.

You can access the Porting Assistant for .NET Visual Studio IDE extension from the Visual Studio Extensions Marketplace. Search for "Porting Assistant for .NET". After you download and install the extension, you can assess any solution in your portfolio.

Porting Assistant for .NET Visual Studio IDE extension topics

- Supported versions (p. 14)
- Prerequisites for using the Porting Assistant for .NET Visual Studio IDE extension (p. 15)
- Memory requirements (p. 15)
- Pricing for the Porting Assistant for .NET Visual Studio IDE extension (p. 15)
- How Porting Assistant for .NET Visual Studio IDE extension works (p. 15)
- Install Porting Assistant for .NET Visual Studio IDE extension (p. 17)
- Assess and analyze solution using Porting Assistant for .NET Visual Studio IDE extension (p. 17)
- Port solution using Porting Assistant for .NET Visual Studio IDE extension (p. 18)
- Transition from standalone tool to Porting Assistant for .NET Visual Studio IDE extension (p. 18)
- Troubleshoot the Porting Assistant for .NET Visual Studio IDE extension (p. 19)
- Porting Assistant for .NET Visual Studio IDE extension version history (p. 14)

Supported versions

The Porting Assistant for .NET Visual Studio IDE extension supports the following .NET versions:

- Source versions: .NET Framework 3.5 and later
- Target versions: .NET Core 3.1 and 5

For assessments and porting, both Windows services and ASP.NET applications are supported.

The Porting Assistant for .NET Visual Studio IDE extension supports the following versions of Visual Studio and Visual Studio Code:

- Visual Studio 2019 and later

  Note
  If you have Visual Studio IntelliCode installed, you must change the C# suggestions to **Disabled**. Select **Tools** > **Options** > **IntelliCode** > **Suggestions** > **C#**, then select **Disabled** from the dropdown.

Prerequisites for using the Porting Assistant for .NET Visual Studio IDE extension

To use the Porting Assistant for .NET IDE extension, ensure the following prerequisites:

- Installation of .NET Core 3.1 or 5. Download .NET Core.
- **AWS CLI**: You must have a valid AWS CLI profile in order for Porting Assistant for .NET to collect compatibility information on the public NuGet packages and the APIs within the packages that are in use by your application. To view the type of application data collected by Porting Assistant for .NET , see Data collected by Porting Assistant for .NET (p. 21). Information about public NuGet packages is collected to help AWS prioritize work to address .NET Core incompatibilities on the NuGet packages, if any. For instructions on how to configure an AWS CLI profile, see Configuring the AWS CLI .
- Visual Studio 2019 version 16.9 and later.
- Installation of any .NET Framework versions currently in use by your application.

Memory requirements

The following memory requirements must be met to use the Porting Assistant for .NET Visual Studio for .NET IDE Extension.

Solution size	Minimum memory requirements
Small solutions (1,000 to 50,000 lines of code)	4 GB
Medium solutions (50,000 to 400,000 lines of code)	8 GB
Large solutions (400,000 or more lines of code)	16 GB or more, depending on size of source code

Note
These requirements are provided as estimates. Individual solutions can vary for memory usage.

Pricing for the Porting Assistant for .NET Visual Studio IDE extension

The Porting Assistant for .NET Visual Studio IDE extension is available for use at no cost.

How Porting Assistant for .NET Visual Studio IDE extension works

You can download the Porting Assistant for .NET Visual Studio IDE extension from the Visual Studio Extensions Marketplace. When you use the extensions from within a supported Visual Studio version, you can initiate one or more .NET application assessments and port your solutions to .NET Core for

Linux. Additional work may be required to address issues for which Porting Assistant for .NET can't find resolution. You can browse the source code repository to find and specify a solution file for your .NET application, and then initiate an assessment task for the application. From within Visual Studio, you can view and analyze the generated report. When an assessment is complete, Porting Assistant for .NET jump-starts the process by converting .csproj files from the Visual Studio 2015 to the Visual Studio 2019 format. It converts the project file structure to match the structure expected by .NET Core.

You can use the updated project files to start making changes in the source code and to receive contextual help and authoritative guidance from within the editor. The Porting Assistant for .NET Visual Studio IDE extension supports code diagnostics. When an application is ported, you can utilized the AWS toolkit for Visual Studio to deploy it to your choice of AWS .NET Core-supported execution environment.

You can continue to use the standalone Porting Assistant for .NET tool to perform portfolio assessments. You can also begin your assessment with the standalone tool and transition to the IDE environment to port your application.

The following table compares features of the standalone tool and the IDE extension.

Features	**Porting Assistant for .NET standalone tool**	**Porting Assistant for .NET Visual Studio IDE extension**
Run assessment on all applications in a portfolio	Run assessments on multiple applications concurrently	Run assessments only on currently opened applications
View list of assessed applications and compare assessments for all applications	Yes	No
Load single solution to tool for assessment	Yes	Yes
View list of incompatibilities for a solution	Yes	Yes
View package dependency tree for a solution	Yes	No
View recommendations	Yes	Yes
Run automated porting	Yes	Yes
Open source file for direct editing	No	Yes
Contextual help with recommendations	No	Yes
Provide details about compatibility and steps to rectify incompatibilities	No	Yes
Continuous reassessment as code is modified	No	Yes
Test, debug, and run code	No	Yes
Deploy the application in AS runtimes, using AWS toolkit downloaded separately	No	Yes

Install Porting Assistant for .NET Visual Studio IDE extension

Perform the following steps to install the Porting Assistant for .NET Visual Studio IDE extension.

1. Go to the Visual Studio Extensions Marketplace. Search for "Porting Assistant for .NET". Choose **Install** and run the installer to complete the installation.
2. When the installation is complete, **Close** the installer and relaunch Visual Studio. When you relaunch Visual Studio, open the solution you want to assess, then open one of its .cs files. Select the **Analyze** tab to view Porting Assistant for .NET options. To run an assessment, select **Analyze**>**Run Full Assessment**. The first time you run an assessment, you will see the **Getting started with Porting Assistant for .NET for Visual Studio** screen. Enter your AWS named profile details and consent to share telemetry to continue. You must have a valid AWS CLI profile in order for Porting Assistant for .NET to collect compatibility information on the public NuGet packages and the APIs within the packages that are in use by your application. To view the type of application data collected by Porting Assistant for .NET , see Data collected by Porting Assistant for .NET (p. 21). Information about public NuGet packages is collected to help AWS prioritize work to address .NET Core incompatibilities on the NuGet packages, if any. For instructions on how to configure an AWS CLI profile, see Configuring the AWS CLI .
3. From the **Getting Started** screen, you can choose the **Target Framework** version to which you want to port your application.
4. **Save and close** the **Getting Started** screen.

   You can change the settings you entered into the **Getting Started** screen by selecting the **Tools** tab, and choosing **Options** from the dropdown menu. Under **PortingAssistant Extension**, select **Data usage sharing** to select a different **AWS Named Profile**, **Add a named profile**, or to change your usage data selection. Choose **General** under **PortingAssistant Extension** to update the **Target Framework**.

Assess and analyze solution using Porting Assistant for .NET Visual Studio IDE extension

To assess your solution and analyze the results from the Porting Assistant for .NET Visual Studio IDE extension, perform the following steps:

1. Open a solution file, then open a .cs file within the solution.
2. From the Porting Assistant for .NET Visual Studio IDE Extension, select the **Analyze** tab. In the drop-down menu, you can choose to **Enable Incremental Assessments** or **Run Full Assessment**.

   a. **Enable Incremental Assessments**. When you select this option, Porting Assistant for .NET automatically runs a continuous assessment as you make changes to the source code. Compatibility errors are displayed when encountered.

   b. **Run Full Assessment**. When you select this option, Porting Assistant for .NET runs a one-time, full assessment for the compatibility solution loaded in the IDE.
3. The **Error List** pane at the bottom of the screen displays all of the incompatibilities discovered in the source files associated with the solution. You can select each entry in the list of incompatibilities to view the incompatibility in the source code, which is highlighted.

Port solution using Porting Assistant for .NET Visual Studio IDE extension

The following functionality is available when you port your solution using the IDE extension.

Guided assistance

You can fix all of the code recommendations at one time by porting the project, or you can go through each issue within the source code by viewing the code next to each light bulb. When you select to port a project or solution, Porting Assistant for .NET opens a dialog box that prompts you to choose whether to apply recommended source code changes. If you do not select this option, you can go through each issue within the source code. When you make a code change, you must save your updates to view the updated recommendations for your changes.

When you port your solution to .NET Core, the Porting Assistant for .NET Visual Studio IDE extension provides guided assistance in the source editor throughout the process. You can choose to **Port Solution to .NET Core** (all projects) or **Port Project to .NET Core** (single project) from either the **Project** tab, the **Extensions** tab, or the **Solution Explorer** pane to the right of the source file. When the source file is open with a source editor, porting options are provided to assist you, and to automate the porting of each source file. Automated porting involves running the source code through code translation assistant rules for porting. This can be performed at the solution level when you select **Port to .NET Core** and choose to apply recommended source code changes.

View incompatibilities

You can view the list of incompatibilities, suggestions, and more from the **Error list** pane at the bottom of the page. Select an error in the list to go to the section in the source editor where the error is located. The source file will be annotated with the error message, details about the compatibility issue in the annotated line of code, and a recommended solution. These details are included in the code comments. If you have already completed automated porting, an annotation at each (denoted by a light bulb) line is modified to describe the change. The source file also highlights each line of source code with compatibility issues. This helps you to visualize the magnitude of the issues in the file and to begin to address them. The highlighting is removed when the compatibility issues are addressed.

Suggestions

The source editor provides a replacement suggestion for each incompatibility in the source file. If a direct replacement exists, you are given a choice to select and replace it. If there is no direct replacement, references or contextual help on how to proceed are provided.

Run code

When all of the source files are updated, you can compile, test, and run your code. If you have the AWS Toolkit for Visual Studio plug-in installed on your IDE, you can deploy your application on Amazon Web Services. To install, see AWS Toolkit for Visual Studio Code.

Transition from standalone tool to Porting Assistant for .NET Visual Studio IDE extension

> **Note**
> If the Porting Assistant for .NET Visual Studio IDE extension is not installed on Visual Studio, the transition process will include steps to download and install the extension.

From the **Assessed Solutions** page of the standalone assessment tool, choose **Open the solution in IDE**. Visual Studio will open the solution and assessment that you performed using the standalone tool. Once

the IDE extension opens, you can port the solution with contextual assistance. If your solution was at the assessment stage, all of the assessment details are shared with the IDE extension. If you started the automated porting of the solution in the standalone tool, the IDE loads the current state of porting.

Troubleshoot the Porting Assistant for .NET Visual Studio IDE extension

If the Porting Assistant for .NET extension is disabled, contact aws-porting-assistant-support@amazon.com.

Porting Assistant for .NET Visual Studio IDE extension version history

The following table describes the released versions of Porting Assistant for .NET Visual Studio IDE extension.

Version	Details	Release date
1.0.3	• Adds memory logging. • Fixes log uploader. • Adds default upgrade version of NuGet packages in porting recommendation. • Syncs metric types with Porting Assistant standalone tools. • Adds default porting package version in the porting request.	August 13, 2021
1.0	Initial release	May 3, 2021

Security in Porting Assistant for .NET

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Porting Assistant for .NET, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Porting Assistant for .NET. The following topics show you how to configure Porting Assistant for .NET to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Porting Assistant for .NET resources.

Topics

- Data protection in Porting Assistant for .NET (p. 20)
- Identity and Access Management for Porting Assistant for .NET (p. 21)
- Configuration and vulnerability analysis in Porting Assistant for .NET (p. 22)
- Security best practices for Porting Assistant for .NET (p. 22)

Data protection in Porting Assistant for .NET

The AWS shared responsibility model applies to data protection in Porting Assistant for .NET. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form fields such as a **Name** field. This includes when you work with Porting Assistant or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Topics

- Data collected by Porting Assistant for .NET (p. 21)

Data collected by Porting Assistant for .NET

If you accept the data collection option in the **Settings** menu of the Porting Assistant for .NET tool, the following application data is collected:

1. Application errors generated when running assessments, porting, or when performing other functions provided by the Porting Assistant for .NET tool.
2. Names and versions of public NuGet packages assessed by the Porting Assistant for .NET tool.
3. Metrics for assessments run by the Porting Assistant for .NET tool on public NuGet packages, such as the number of packages and solutions, and the amount of time taken to create a solution.

You can change your data collection settings at any time in the **Settings** menu of the Porting Assistant for .NET tool.

Encryption at rest

All data within Porting Assistant for .NET is encrypted at rest in accordance with industry standards.

Encryption in transit

All requests to Porting Assistant for .NET must be made over the Transport Layer Security protocol (TLS). We recommend TLS 1.2 or later.

Identity and Access Management for Porting Assistant for .NET

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. Porting Assistant for .NET is a standalone application that does not require IAM access control to use resources.

To use Porting Assistant for .NET, you must attach the following IAM policy as an inline policy to your IAM user. Then, configure a profile on your server with the IAM credentials of this user. For steps on how to attach this policy, see AWS Identity and Access Management (IAM) (p. 7).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "EnCorePermission",
            "Effect": "Allow",
            "Action": [
                "execute-api:invoke",
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:492443789615:3dmmp07yx6/*",
                "arn:aws:execute-api:us-east-1:547614552430:8q2itpfg51/*",
                "arn:aws:s3:::aws.portingassistant.dotnet.datastore",
                "arn:aws:s3:::aws.portingassistant.dotnet.datastore/*"
            ]
        }
    ]
}
```

Configuration and vulnerability analysis in Porting Assistant for .NET

When the Porting Assistant for .NET requires updates, you are notified and will be required to install the latest version of the application upon restart. You maintain the system patching responsibility, per the shared responsibility model.

Security best practices for Porting Assistant for .NET

Porting Assistant for .NET provides security features to consider as you develop and implement your own security policies. The following best practices are general guidelines and don't represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful considerations rather than prescriptions.

Implement least privilege access

When you attach the IAM policy (p. 7) as an inline policy to your IAM user, grant only the permissions that are required to perform the specified task. Implementing least privilege access is fundamental in reducing security risk and the impact that could result from errors or malicious intent.

Metrics collected by AWS Porting Assistant for .NET

Porting Assistant for .NET collects the metrics and logs listed in the following location of the server/desktop where the tool is run: `C:\Users\username\AppData\Roaming\Porting Assistant for .NET\logs`.

Porting Assistant for .NET version history

The following table describes the released versions of Porting Assistant for .NET.

Version	Details	Release date
1.5.3	• Adds memory logging. • Moves logging to a separate thread to avoid assessment interference. • Adds prerequisite checks (internet and file access) before running assessment.	July 22, 2021
1.5.2	**Bug fixes** • Fixes issue with hanging assessment jobs. • Fixes broken links. • Adds support for error notification when builds fail. • Dependency versions updated.	June 13, 2021
1.4.3	**Bug fixes** • Mitigates null reference issue when performing assessment. • Fixes missing files when porting to new location. • Adds support for filtering failed project when porting.	March 26, 2021
1.4.2	**Bug fixes** • Fixes sorting issue for project tables. • Matches projects with corresponding upgraded packages when porting whole solution. • New configuration window appears when using an unsupported version. • Added new API metrics.	March 18, 2021
1.4.1	• Adds Porting Assistant client version track support. • Adds option to port entire solution. • Adds support for displaying number of porting actions. • Adds support for displaying only released NuGet versions. • Adds Owin rules for namespaces: Owin, Microsoft.Owin, and Microsoft.Owin.Hosting. • Adds Owin support for new rule action type to replace method modifiers. • Adds Owin support for new rules to remove base class of OwinMiddleware on custom pipeline objects and remove the override method modifiers on the invoke method. • Adds Owin support for unit tests for all of the new rules listed previously.	February 9, 2021

Version	Details	Release date
	• Fixes blank page on external link navigation. • Fixes bug causing porting project file failure when packages are selected.	
1.4.0	• Adds recommendation rules for OWIN. • Adds .NET Standard 2.1 support to applicable rules. • Updates config migration action to process config files more reliably. • Bug fixes for the code translation assistant feature.	January 15, 2021
1.3.0	• Adds automatic code translation feature. • Displays additional recommended code changes in the source file view. • Performs some recommended changes on source code when porting. • Adds more details to exported .csv files when exporting APIs.	December 16, 2020
1.2.0	• Adds support for .NET 5.0. • Adds .NET 5.0-related packages and API data in DataStore.	November 25, 2020
1.1.0	Transition to use open sourced Porting Assistant for .NET Client (https://github.com/aws/porting-assistant-dotnet-client).	October 15, 2020
1.0.2	• Adds EULA and metrics collection details to Settings page. • Fixes permissions issue that prevented app from querying compatibility data files. • Includes Porting Assistant version information in calls to backend. • Improves compatibility accuracy of API calls from SDK.	October 5, 2020
1.0.1	• Displays API and Source table empty states. • Fixes an error in telemetry that causes null pointer exception in some cases. • Fixes telemetry issues that prevent buffers from flushing at time intervals. • Fixes flash bar not appearing in certain race conditions. • Adds solution as a dimension in metrics and logs. • Adds frontend logs as part of returned logs. • Fixes an issue that causes API analysis to improperly cache.	July 29, 2020
1.0.0	Initial release	June 30, 2020

Document History for Porting Assistant for .NET User Guide

The following table describes the documentation for this release of Porting Assistant for .NET.

- **Latest documentation update:** October 1, 2020

update-history-change	update-history-description	update-history-date
Updated Porting Assistant for .NET required IAM policy. (p. 26)	Updated the IAM policy that you must attach as an inline policy to your IAM user.	October 1, 2020
Porting Assistant for .NET initial release (p. 26)	Initial release of Porting Assistant for .NET: a tool that helps you to port your existing .NET applications running on Windows Server to .NET Core on Linux.	June 30, 2020