FAQ: Support for ASP.NET Core / .NET Core / .NET 5 / 6 / 7 / 8+

IMPORTANT: NEW FOR 11.4

Up until version 11.4, the original answer to "do you support .NET Core" was "In a very limited way - we support ASP.NET Core on web apps but targeting .NET framework"

With the release of 11.4 we have finally been able to bring .NET (.NET 6 and up) into DotImage

The older means of using ASP.NET Core targeting .NET framework are still technically valid for 11.3 and older, but going forward, when moving to 11.4 you will need to re-target to use .NET 6 and up (technically .NET 5 is possible but since MS has ended support for .NET 5, we consider it not officially supported)

PLEASE DO NOT use the old 11.3 or older (ASP.NET Core targeting .NET Framework) for any new web development as this was only a stop gap workaround until we could get proper .NET 5/ .NET 6 support running

NOTE we do not have support for .NET Core 3.1 at all

IMPORTANT: New for 11.5

In 11.5, we dropped the 5.0 dlls and now compile against .NET 6 which is the oldest officially supported .NET Runtime (we don't support .NET Core / .NET Standard so those are not available)

When .NET 6 ends life / support, the version that comes out after that (11.6) will target .NET 8 and change to 8.0 dlls

Cross Platform (Linux) is NOT supported

Unfortunately, in order to support true cross platform deployment (such as is needed to deploy to Linux / Linux containers) our SDK would need to use "fully managed code". We have a huge amount of assembly language and C++ at the heart of our internal code and such an endeavor would require rewriting / porting ALL of it.

Thus our .NET 5 / .NET 6 / .NET 7 / .NET 8 support can only be used on Windows containers/hosting

Locations of relevant resources

Desktop / Winforms / Console Apps (non-web)

If you're looking to target desktop / console apps, etc, you can find our DLLs for use in .NET 5 and up on your development machine where you installed DotImage 11.4

They can be found in

C:\Program Files (x86)\Atalasoft\DotImage 11.4\bin\5.0\x86

and

C:\Program Files (x86)\Atalasoft\DotImage 11.4\bin\5.0\x64

For a complete tutorial on how to build a console app in .NET 6 with DotImage, please see:

INFO: .NET 6 Console App Whitepaper - Getting Started with .NET 6

Web Application Development

For web components it is critical that you use the proper NuGet packages .. our extensive tutorial on how to get up and running with .NET 6 covers the details but the quick start / TL;DR: is:

Use the NuGet package console and issue this command:

```
nstall-Package Atalasoft.DotImage.WebControls.Core.x64
```

For a complete tutorial on on how to use our WebDocumentViewer in .NET 6 please see:

INFO:WDV (and WebCapture) In .NET 6 (.NET Core) Whitepaper - Getting Started

IMPORTANT NOTE about .NET Core and WinForms / WPF Designers

UPDATED July 2023

Unfortunately, our .NET 5 / .NET 6 WinForms controls (Viewers and other visual components that are generally added via the designer) components do not work with the **DESIGNER**. This is due to a known issue in Visual studio handling of .NET5 and custom controls. We are currently investigating if /when the issue is addressed in .NET 6 and hope to be able to offer full support for WinForms and WPF designer mode when targeting .NET 6 in the future

HOWEVER, we have found it is possible to use our viewer controls in a .NET 6 app by instantiating the object at runtime. This means you can't drag /drop to the designer and work with the properties grid the way you can in .NET framework but we've found you can instantiate the viewer object at runtime and add it to a form's Conrols collection or to the Controls collection of any valid container that takes Controls

Simple example: you have a new WinForms project with just form1 and you want our viewer to be the entire contents of the form (maybe you added a MenuStrip and StatusStrip but you want to put the viewer as the only control

In the class containing, it you'd add a field for the viewer type - lets say

private WorkspaceViewer _viewer;

In the constructor for form1 after the InitializeComponent(); you'd do something like this

```
sing Atalasoft.Imaging.WinControls; public partial class Form1 : Form { WorkspaceViewer
_viewer; public Form1() { InitializeComponent(); _viewer = new WorkspaceViewer();
_viewer.Dock = DockStyle.Fill; // set any other _viewer.Property // bind to any needed
_viewer.Event etc... this.Controls.Add(_viewer); } }
```

Now, you can use the viewer in events/methods etc such as a button click to open a new image in the viewer:

```
rivate void fileOpen_Click(object sender, EventArgs e) { using (OpenFileDialog dlg = new
```

```
OpenFileDialog()) { if (dlg.ShowDialog() == DialogResult.OK) { _viewer.Image = new
AtalaImage(dlg.FileName); _viewer.Refresh(); } } }
```

Or, maybe apply a command like a rotate:

```
rivate void rotateRight_Click(object sender, EventArgs e) { RotateCommand rot = new
RotateCommand(90); _viewer.ApplyCommand(rot); }
```

We have attached a working sample .NET 6 WinForms app that uses our WorkspaceViewer, and implements open, save and a couple of commands. Its attached to this KB article as AtalasoftNet6SampleWinforms.zip

NOTE: that it is NOT possible to take an existing app that uses our controls in designer in .NET framework and "Add Existing" or otherwise just drag it into a .NET 6 WinForms app. There are several breaking changes from .NET 6 that make this not work even for forms not including Atalasoft components. You will need to make a new form and redo the design (minus the Atalasoft designer, using the above instead)

For the time being, if you really want to use the designer or don't want to have to redo all your forms , just continue .NET Framework (we support 4.5.2 up through 4.x) instead of .NET 5 / .NET 6 / .NET 7 / .NET 8

.NET Core 3.1 / .NET 5 / .NET 6 / .NET 7 / .NET 8 (sometimes referred to as .NET Core or .NET Core Native)

A lot of customers have asked about .NET Core 3.1 / .NET 5 / .NET 6 / .NET 7 / .NET 8 / .NET Core support.

Up to and through 11.3 DotImage has been completely and only a .NET Framework SDK .. in the past, before .NET Core, that was clear enough as the only ".NET" was shorthand for .NET Framework.

Now with Microsoft .NET Core and .NET Framework and ASP.NET Core, there can be

confusion.

When referring to 11.3 and older, our SDK only a .NET Framework SDK, and unfortunately, at this time, the answer is no, we do not support .NET 5 / .NET 6 / .NET 7 / .NET 8 / .NET Core Native

For .NET 5 and up, please move to DotImage 11.4

DotImage 11.4 ships with DLLS for 5.0 which work for .NET 5, .NET 6, .NET 7, and .NET 8

LEGACY RESPONSE REGARDING 11.3 AND OLDER - OUT OF DATE INFO

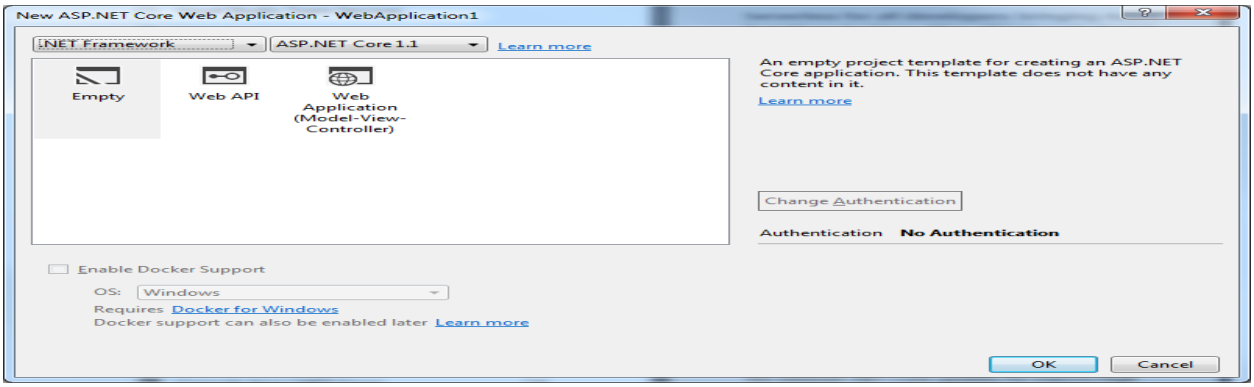
Support for ASP.NET Core Apps for WebDocumentViewer and WebCapture

As of 11.0, we now do have support for ASP.NET Core apps targeting the .NET framework for our WebDocumentViewer

IMPORTANT NOTE:

Atalasoft DotImage components can be used along with ASP.Net Core libraries, but it requires the application to be built for .Net Framework 4.5.2 for ASP.NET Core 1.x and at least Framework 4.6.2 for ASP.NET Core 2.x

You should create new projects in Visual Studio from the ASP.NET Core Web Application template, using .NET Framework and ASP.NET Core 1.1 like this:



By default the left combobox has value .NET Core which we do not support. If you try to create new project with default settings, you won't be able to use the WDV.

NOTE: we do support targeting ASP.NET core 2.x .. version 11.0 works with it, but you will need to target at least .NET framework 4.6.2

We have a similar screenshot in our tutorial, but Microsoft updated its template in a recent Visual Studio update.

Customizing Middleware / Event Handling (Formerly Custom Handler)

In the tutorial, you will see a brief mention about customizing the middleware:

If you want to customize this middleware, you can choose another overload of the method RunWebDocumentViewerMiddleware that accepts implementation of the interface IWebDocumentViewerCallbacks.

However, it's a bit light on details. So here's the "missing piece"

Instead of this line

```
app.Map("/wdv", wdvApp => { wdvApp.RunWebDocumentViewerMiddleware(); });
```

use this line:

```
app.Map("/wdv", wdvApp => { wdvApp.RunWebDocumentViewerMiddleware(new MyWDVCallbacks()); } );
```

and add this class

```
ublic class MyWDVCallbacks : WebDocumentViewerCallbacks { public override void
DocumentInfoRequested(DocumentInfoRequestedEventArgs args) {
Console.WriteLine("**********===DocumentInfoRequested===**********");
base.DocumentInfoRequested(args); } public override void
ImageRequested(ImageRequestedEventArgs args) {
Console.WriteLine("**********===ImageRequested===**********"); base.ImageRequested(args); }
```

```
/* * The above was just an example of hooking to the two main events.. you can handle any exposed event here * * //.. most common ... * AnnotationDataRequested * DocumentSave * DocumentStreamWritten * AnnotationStreamWritten * //.. less common . * PageTextRequested * PdfFormRequested * //.. rarely if ever used .. * ReleaseDocumentStream * ReleasePageStream * ResolveDocumentUri * ResolvePageUri */ }
```

Support for ASP.NET Core 2.x

The 11.0 and 11.1 versions were only certified for ASP.NET Core 1.x (Targeting .NET Framework 4.5.2 or higher). It was possible to target ASP.NET Core 2.x, but was not officially certified.

The 11.2 version now officially supports ASP.NET Core 2.1 targeting .NET Framework 4.6.1 or higher.

Support for ASP.NET Core 3.x

At this time we do not have certified support for ASP.NET Core 3.x though you may find that you can target it with the latest .NET Framework - keep an eye on release notes for later versions.

Support for ASP.NET Core in Legacy Web Controls

Our legacy web controls (WebImageViewer / WebAnnotationViewer / WebThumbnailViewer) do NOT have any support for ASP.NET Core or .NET Core native. The only support for ASP.NET Core we provide at this time is for the new WebDocumentViewer / WebDocumentThumbnailer controls mentioned above.

ASP.NET Core Blazor

The question of whether we support ASP.NET Core Blazor has come up. At this time (as of 11.4) we do not officially support it.

Indications are that it may be possible to use jQuery and js within the Blazor pages, and in theory if they let you run them and you can load our required js and css files and reference

the required jQuery, jQuery-UI, etc, it might be able to work.

Please see https://www.talkingdotnet.com/using-jquery-with-asp-net-core-blazor/

However, at this time it is not tested and would be considered "use at your own risk". We require an IIS server on the back end running ASP.NET or ASP.NET Core (targeting .NET Framework) for our handler / middle-wear to run. There is no client-only option.

More Resources

INFO:WDV (and WebCapture) In .NET 6 (.NET Core) Whitepaper - Getting Started

Original Article:

Q10467 - FAQ: Support for ASP.NET Core / .NET Core

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50005/FAQ-Support-for-ASPNET-Core-NET-Core...