Accelerating your .NET update: A playbook

A planned approach to adopting .NET latest versions and staying ahead of the curve
Are you thinking about upgrading to the newest version of .NET? Perhaps your business is operating on an older .NET Framework or .NET Core version, and you’re considering migrating your software. Even if you haven’t considered upgrading yet but are curious about the benefits of the recent versions of .NET, this guide will provide you with all the knowledge you need.

With .NET 6, released in November 2021, Microsoft has created a single distribution, open-source framework that enhances the developer experience. .NET 6 completes the unification of Microsoft’s .NET ecosystem and offers unparalleled execution speeds, memory usage, and team productivity that will directly affect your bottom line.

.NET 6 is truly the best option available at the moment, and while .NET 7 is scheduled to arrive at the end of 2022, .NET 6 is the long-term support version your software should target. This guide will walk you through the changes brought with .NET 6, the business benefits of upgrading, and the process of migrating. Because upgrading to .NET 6 is a software leap for most companies, partnering with .NET experts can save you time, money, and stress.

The evolution of .NET and what it means for your business

.NET is a cross-platform, open-source developer platform designed for building multiple types of applications. The original iteration, .NET Framework, supports websites, services, desktop apps, and more, on Windows. Xamarin/Mono is a .NET implementation for running apps on all the major mobile operating systems. .NET Core, introduced in 2016, became the cross-platform successor of .NET Framework, supporting Windows, Linux, and Mac operating systems. However, with the release of .NET 5 in 2020, Microsoft decided to unify .NET Framework, .NET Core, and Xamarin into a single “.NET.” This was a huge step in the evolution of the ecosystem. The latest release, .NET 6, promises major leaps in developer productivity and execution speeds. As reviews state, “there is virtually nothing you can’t do with .NET 6 which would require you to tether yourself to the aging .NET Framework.”

Microsoft, in fact, is deprecating .NET Framework, with version 4.8, released in 2019, being its last.

Beginning with the release of .NET 6 in November 2021, Microsoft has committed to releasing a new version annually. Each subsequent update has brought notable performance improvements, productivity gains, improved execution speeds, and improved memory usage critical to software builds in the .NET ecosystem. With every new version, Microsoft will continue to build upon improvements made in previous versions. As a result, if you opt to wait for another version before you upgrade, the migration process will be far more complex. Also, since only even versions will have long-term support, upgrading to .NET 6 ensures the benefit of full-term support.

If you are currently on .NET Framework or any version of .NET Core, upgrading to the newest version of .NET is the wisest option. Not only will you benefit from the execution speed and productivity benefits .NET 6 offers, you’ll put yourself in the best position to continue evolving along with the platform.

With .NET, the benefits and changes create ample opportunity for increased productivity and software improvements. The implementation of class libraries is unified, providing two options of runtimes for users, CoreCLR and Mono. Both options run within .NET, with CoreCLR providing a high-throughput, high-performance runtime at a high level and Mono providing a lightweight (but not as fast) runtime. Unlike with prior versions of .NET, runtime choices now complement the workloads historically used most — CoreCLR, for server and desktop applications, and Mono for mobile and lightweight uses, such as WebAssembly. .NET also features a unified runtime that can implement both C# and F# code across all platforms, further increasing productivity.

Benefits of upgrading to .NET 6

- Flexible project deployment
- Unparalleled performance
- Reduced infrastructure and hosting costs
- Modern, innovative languages and APIs
- Unified platforms and SDKs
- Modernized .NET codebases
- Dynamic profile-guided optimization (PGO) system
Why upgrading to .NET is critical

Migrating to .NET is critical because support for any version released prior to .NET Core 3.1 will be phased out. Microsoft's new support policy promises that any new odd-numbered versions (e.g., .NET 5) have patch support for 18 months, and new even-numbered versions have patch support for three years (e.g., .NET 6). Microsoft is retiring support for .NET Framework versions 4.5.2, 4.6, and 4.61 in April 2022 and .NET Core 3.1 in December 2022. Any previous versions are already out of the support cycle. Whether part of a modernization effort or simply to sustain your current software, migrating to .NET will benefit both the business and tech teams. Upgrading to .NET 6 now ensures you will benefit from the full-term support provided by Microsoft. This is important even if you don’t think you’ll have customer support on speed dial—the point is that .NET 6 will gain security fixes over the support window whereas older versions will fall out of the fixes window.

Furthermore, the updated .NET features are impressive, especially if you’re still on an older .NET Framework or .NET version. .NET provides a far superior user experience than previously available. .NET 6 creates substantial performance improvements and productivity gains, critical to scalability and supportability. Subsequent version releases should prove to continue to improve the speed and capabilities within the .NET ecosystem.

Migrating sooner rather than later ensures your team benefits from current .NET improvements and will continue to reap future benefits.

Most significant changes in .NET 6

- Provides two options of runtimes for users: the high-performance CoreCLR (for server and desktop applications), and the lightweight Mono (for mobile and WebAssembly)
- Has binary libraries that work on all the various environments as well as a common surface of APIs that work across all versions of .NET
- Users have a choice of a runtime (CoreCLR or Mono) and compilation system (static compilation, JIT compilation, tiered, or interpreted)
- Developers writing apps have access to the latest C# version and features, now v10.0 with .NET 6
- Fully cross-platform and compatible across Windows, Mac, and Linux operating systems
- Enables anyone building applications to take advantage of a cloud computing architecture for improvement in performance
True .NET worth: The business benefits of upgrading

All modernization and migration projects take time and resources, but you have to counter these against the savings you’ll incur once your software’s truly up to speed. Beyond that, the new processes you put in place along the way, along with the embracing of change, can fortify your teams and your business for the long term. These are just a few on the rewards you can expect to reap by upgrading to the latest .NET version:

**Less troubleshooting**

Migrating to the newest version of .NET ensures the most extended support lifetime offered which means you can count on swift solutions and improved project focus. .NET 6 specifically offers a three-year support window critical to long-term efficiency. The three-year support cycle establishes a greater return on investment and ensures your software will have three years of scalability.

**Increased output**

The performance improvements .NET 6 brings for your developers translate to greater team productivity and faster time to market. Not only does .NET 6 improve productivity, but you may also see a significant boost in your team’s morale, as well as in the value they are delivering to customers, thanks to new features that improve workflows and promote efficiency.

For example, the Razor compiler has received drastic speed improvements that Microsoft benchmarks show decreased build times up to 50% (seen in ASP.NET 6). This dramatic speed improvement can add up when working with large projects and allows for more effective and efficient maintenance and support functions.

In addition, the hot reload feature accelerates the adjustment/impact loop for developers. A new reality for ASP.NET apps in C#, the hot reload enables code edits while the app is running. Thus, changes are automatically reflected in the app without losing the app state – a nice improvement for debugging and interactive development workflows.

**Improved costs**

Remaining with .NET Framework or an older version of .NET Core has hard costs, both in terms of people, and

---

2https://devblogs.microsoft.com/dotnet/announcing-net-6/
actual cash spent long-term. The older the technology is, the harder it is to find employees or contractors who can work with it. Even when you find someone, you may end up paying a premium for that niche knowledge—one which only grows the longer you delay. Otherwise, you will need to have substantial training programs in place to train new programmers on your legacy frameworks, increasing the possibility of errors as employees onboard. Legacy frameworks also take more server space and cost more to operate daily. They also tend to incur higher maintenance costs, which include high support and licensing costs.

**Win with speed**

Our mantra as a software company has always been “Win with speed,” and Microsoft has undoubtedly given the developer community the right tools to do so. Based on the improvements with .NET 5, and now .NET 6, the indications are that subsequent versions will bring more speed, more efficiency, and more productivity to teams. This impacts supportability and scalability as updates, new releases, and bug fixes can be accomplished faster and more affordably. Upgrading now to .NET 6 will enable you to scale your apps faster than previously possible.

## Preparing to migrate to .NET: Key considerations

Whether simply modernizing your software or looking to get disparate apps to work together in .NET, facilitating a proper migration to .NET is crucial. Before your organization begins the migration process, the team should first conduct a code analysis to determine how compatible your solution is with the .NET version you’re targeting. Microsoft has developed a Portability Analyzer tool³ that will analyze your code and determine the compatibility of your current frameworks with the new .NET target framework.

**Mapping your path**

Once you’ve confirmed your target .NET version, it’s critical to understand the migration paths available before you start out. Depending on how your software is developed and released, a direct migration may be an option. However, it is most likely that a direct migration is not possible. Although large sections of your code will ultimately work unchanged, it can be difficult to detect and account for the code that does have to change. Depending on your approach, you may inadvertently find yourself making an incredibly high risk, all-or-nothing upgrade. With care, you can instead approach the problem in low-risk incremental steps.

³https://docs.microsoft.com/en-us/dotnet/standard/analyzers/
Upgrading your system may seem fairly straightforward. However, on an intermediate level, the process can become very complex. Of the challenges listed above, none is more detrimental than ending up in a position where your client-facing software no longer works.

Because there is no ideal scenario or process for migrating from a legacy framework to modern .NET, ensuring the tech team undergoing the migration is fully prepared and has a well-planned execution strategy is essential. If done correctly, the migration process is relatively painless and will result in few to-no bugs. However, if done incorrectly, you will find yourself spending a significant amount of time on costly fixes.

**Finding the right .NET partner**

Working with the right software consultancy to plan and execute your upgrade will secure a successful migration. Going with a less experienced firm might result in issues down the line that require fixes in the future. The goal is to migrate from the current system to .NET with the least number of issues possible, so future updates continue smoothly. If you start with an unsteady foundation with this migration, future updates will result in unforeseen errors that can be difficult to solve.

However, not all firms are created equal. A well-executed migration requires top-notch software consultants who are highly experienced with all Microsoft products. 

**Potential challenges**

- Working in steps that are too large and result in errors
- Reaching a deadlock state in a project
- There is no ideal scenario for migrating to .NET
- Avoiding the instinct to conduct bulk file copies or long-lived branches
- Migrating without interfering with your consumer-facing product
- Finding the right tech consultants for your migration
Our .NET migration process: Iterative development and incremental delivery

Our consulting team takes a skillfully planned approach to .NET migrations. We favor hands-on experiments and evidence-based steps that front-load possible obstacles and accelerate throughput. The general strategy is to take small, incremental steps to minimize risk and keep teams moving forward. The steps are iterative, as they plan for the work of one iteration to be improved upon in subsequent iterations, and incremental because completed work is delivered after each iteration.

However, within each delivery cycle, there can be several iterations that are not released to production. Iterations allow the team to test changes and address any issues that arise in each step of the migration.

Migrating complex applications

Applications consisting of 60 or more folders with shared code are tricky to migrate. Because the codebases of .NET Framework (orange squares) and modern .NET (blue squares) are different, folders from .NET Framework cannot be used directly by folders containing the new .NET code, despite their superficial similarity.

Therefore, if you choose to do a direct migration from .NET Framework to modern .NET, you will have to conduct a complete migration – meaning if one folder is in modern .NET (blue squares) code, all folders will need to be written in the new .NET code to avoid compromising the solution.

This migration will only work if the software is not currently distributed or if the software distribution is shut down temporarily. This is obviously not ideal as it inhibits teams from continuing to develop, test, and implement solutions. Fortunately, any given folder of code can, with care, be made to work for both .NET Framework and modern .NET. This fact opens the door to safe, incremental change. Converting your folders one-by-one ensures that your application continues to work throughout the process. Let’s illustrate this incremental migration.

First, we'll look at your application files in .NET Framework.

Then, we convert all folders to common ground, code that is valid for both old .NET and new .NET.

Finally, simplify each folder to merely target the new .NET.
Phases of the migration process

Our consultants employ proofs, checklists, and shims that provide high confidence in every single step of the process. We utilize commit history to double-check every change made in the process to ensure as flawless a migration as possible. This also allows for quicker changes when unforeseen issues are uncovered.

<table>
<thead>
<tr>
<th>PHASE 1</th>
<th>PHASE 2</th>
<th>PHASE 3</th>
<th>PHASE 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identify an incredibly small step that unlocks a plan for the .NET migration.</td>
<td>Create two categories: Code within projects that are easy to migrate (Only need minimal edits) and code that needs a partial or complete re-work.</td>
<td>Organize the real work items into categories such as foundational tasks, cross-complication, and infrastructure to create a critical path of work.</td>
<td>Teams work through the critical path, continually update the process, and address unforeseen errors when they arise through skillful action.</td>
</tr>
</tbody>
</table>

The benefits of a skillfully planned approach

Our iterative and incremental approach supports high team morale from launch to completion. A less experienced consultant may begin the migration process by assigning a programmer the initial task of changing the target framework for the initial project. The programmer would then be met with a barrage of error messages after making the change.

We favor a skillfully planned approach that identifies potential risks and bottlenecks throughout the migration before making any changes. Starting a migration with a waterfall of error messages slows the project down immediately and creates a lull in productivity for the rest of the team. In addition, an instantaneous list of red-marked errors is demoralizing for a programming team. Predetermining error messages enables teams to edit projects simultaneously before changing the target framework on the initial project. This approach effectively lowers the number of errors and creates a clear path to the next step.

The secret to success here is identifying a tiny step that unlocks your .NET migration path, and turning that into a skillful plan. A clear view of critical work provides stakeholders with visibility into scope, dependencies, and progress, encouraging their ongoing participation. The plan can take considerable time to construct and will likely change throughout the process, but the team will be able to adapt to change and pivot flexibly when needed.
Our mantra as a software company has always been “Win with speed,” and Microsoft has undoubtedly given the developer community the right tools to do so.
Why make us your .NET migration partner?

By enlisting Headspring teams to guide your .NET upgrade, you are benefiting from the expertise of a Microsoft Gold Certified Partner. We have earned this designation by demonstrating our expertise and best-in-class capabilities within Microsoft solution areas. Our consulting team is made up of experts with Microsoft products that will provide a seamless and effortless migration experience to save you time and money.

With us, you’re not just getting expert engineers, you’re working with savvy consultants who understand your software, your teams, and your business. We get that the deployability of your software is mission-critical, even during an important upgrade. Every step we take in the migration process ensures that your software is continuously reliable. Your teams can continue to develop and make changes to the solutions throughout the process. Our consultants ensure that the migration process will interrupt your daily tasks as little as possible, because we know customer happiness is the key to success.

Don’t get left in .NET’s dust!

Let our migration experts get you up to speed faster.

Get in Touch