SYNERGEX

Building for the Future with .NET

The strategic move to modernize and strengthen your Synergy solutions

WHITE PAPER

Contents

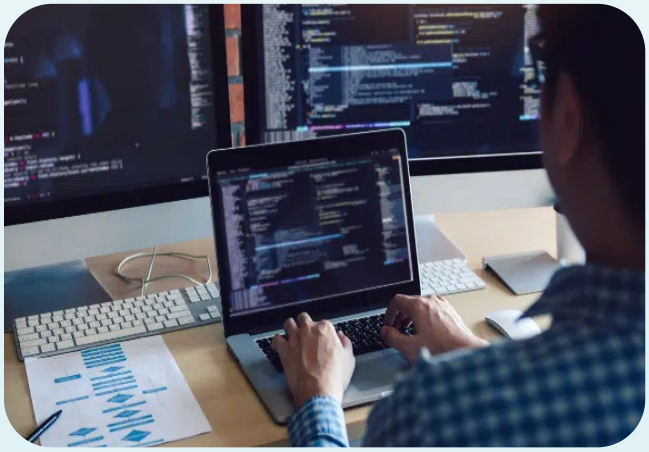
Executive summary

Synergy DBL is a powerful and flexible language designed for developing business-critical applications. As the software development landscape evolves, DBL-based solutions need to keep pace with user demands and new technologies and standards. Synergex's support for DBL applications to run under .NET can offer significant benefits. This white paper explores the advantages of using .NET, the ways it can help developers achieve their goals, and how to get started.

DBL empowers developers

DBL is a proven, high-level programming language that empowers developers to build robust business applications. With its roots in Digital Equipment Corporation's DIBOL language, DBL has evolved to support modern programming paradigms and technologies. It provides

- A versatile and robust language syntax
- Comprehensive, high-performance database management capabilities
- Advanced debugging and profiling tools
- Support for numerous open technologies (including XML, HTTPS, SSL, and JSON)
- Extensive libraries and frameworks for rapid application development



.NET opens opportunities

.NET is a versatile and widely used framework developed by Microsoft that opens new opportunities for software applications. It supports multiple programming languages, libraries, and tools, making it a popular choice for building a wide range of applications. Key features of .NET include

Cross-platform capabilities. Support for Windows and Linux.

Rich libraries. Extensive libraries for networking, data access, distributed computing, and more.

Language interoperability. Ability to use multiple languages (DBL, C#, VB.NET, etc.) within the same project.

High performance. Optimized execution through the Common Language Runtime (CLR).

Scalability. Built-in support for cloud and distributed computing.

Modern development tools. Integration with Visual Studio and Visual Studio Code for enhanced productivity.

DBL and .NET make a powerful pair

Leveraging .NET with your DBL application provides significant benefits.

✔ Enhanced development capabilities

Modern IDE support. By integrating with Visual Studio, developers can take advantage of a state-of-the-art IDE, including IntelliSense, code refactoring, and debugging tools.

Language features. Developers can use modern language features and paradigms available in C# and other .NET languages while maintaining their existing DBL codebase.

Third-party libraries. Access to a vast ecosystem of third-party libraries and NuGet packages significantly enhances development capabilities and reduces time to market.

✔ Improved interoperability

Cross-language compatibility. Developers can create hybrid applications that leverage the strengths of multiple languages, facilitating better code reuse and maintenance.

Web and mobile development. By using .NET, DBL applications can be extended to web and mobile platforms, broadening their reach and usability.

✔ Performance and scalability

Optimized execution. The Common Language Runtime (CLR) optimizes the execution of .NET code, potentially improving the performance of DBL applications when compiled to .NET assemblies.

Asynchronous programming. .NET's asynchronous programming model (async/await) can help build more responsive and scalable applications, especially for I/O-bound operations.

Cloud integration. Native support for cloud platforms like Azure enables easy deployment and scaling of DBL applications in the cloud.

✔ Future proofing

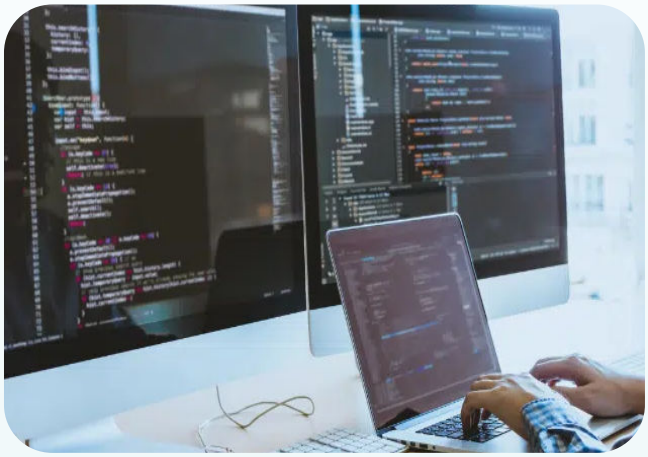
Ongoing support and updates. Microsoft consistently updates and improves the .NET framework, ensuring applications stay up to date with the latest technologies and security standards.

Community and ecosystem. A large and active community provides extensive resources, tutorials, and support for .NET developers, aiding in continuous learning and problem-solving.

Easier developer recruitment and onboarding. Many prospects are skilled in developing for .NET, so the environment will appeal to them. And they can get productive quickly, coding either in DBL or the .NET language they're familiar with.

Cross-platform solutions

Because long-time Synergy developers often have significant investments in UI Toolkit or low-level windows, Synergex has enabled cross-platform windowing on .NET just as it's supported in traditional Synergy. As a result, when running Synergy on .NET 6 or higher, you'll get a consistent functional application experience regardless of whether you're on Windows or Linux. Furthermore, if you compare your traditional application on Linux to one running on .NET, the experience is nearly identical.



You can take almost any traditional Synergy program—whether it's a major one full of business logic or just a utility that needs to perform some extra tasks that are easier in .NET—move it to .NET, and then keep running it as usual. The only difference is that now you're running a .NET assembly instead of a DBR. It doesn't matter if the application is simple with a single screen or a complex mix of windows and console input; everything works the same.

"Moving to Synergy .NET was a game changer for us. If we hadn't made that move when we did, the business wouldn't have grown nearly as fast as it did. We couldn't have been able to produce code as fast as we can now. We were able to get more developers onboarded and contributing faster. Our competition is two to three times bigger than us, but we're able to keep up."

Nate Bahl, Rural Computer Consultants, Inc.

Rural Computer Consultants, Inc.

Same code, same logic, more opportunities

Synergex has made it easier than ever to adopt .NET without substantially reinventing your program or rewriting source code, enabling you to open all your existing business logic and investment to new possibilities. Fundamentally, your application can remain the same: same source code, same business logic. Then, when you want or need to, you can move over any parts of your application that require new functionality. Behind the scenes, your code will be running on a different runtime, but it's all still Synergy.

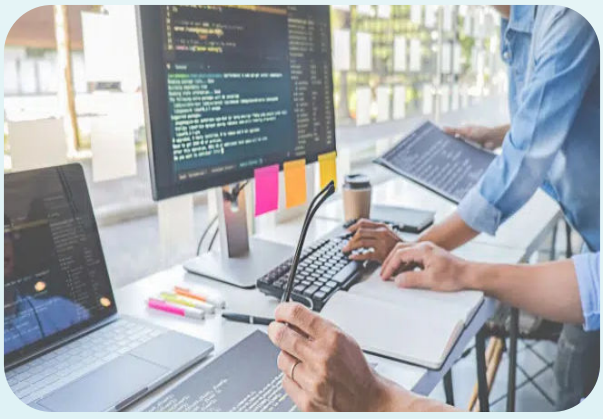
The opportunity for gradual adoption is especially relevant for the typical Synergy developer on Linux who has hundreds of individual Synergy programs that all do their own jobs and call into each other. This web of programs and utilities is ideal for piecemeal conversion. When you suddenly need your report tool to be able to send an email, upload to an API, or interact with some new system that you haven't envisioned yet, you can do it with .NET.

But if the programs are identical in code and function, what advantage is .NET providing? It all boils down to time and value. .NET is a huge ecosystem with broad support from different tool vendors. If you want to integrate with a product, implement an API, or connect to a B2B or C2B system, the odds are excellent that a .NET library already exists for doing it. Thus, you can accomplish your objectives more quickly and effectively.

Example: Updating an email delivery function

Suppose your IT department decides to permanently decommission the SMTP server as part of a transition to a cloud-first IT infrastructure. As a result, corporate policy now mandates the use of SendGrid for all software-related email requirements.

How should you approach this transition? One option is to modify the existing email library by introducing the necessary boilerplate code to support communication with SendGrid's API. This would involve defining data structures for interacting with the API, choosing between the Synergy JSON API or XML API (depending on the required format), implementing the Synergy HTTP document transport API calls, configuring authentication, handling errors, validating responses, and writing appropriate tests. While not an excessively large project, it would require thoughtful planning and could take several days to complete.



However, what if you could accomplish the majority of the SendGrid integration in just a few minutes using .NET?

The first step is to acquire the SendGrid NuGet package. You can do this through the NuGet Package Manager in Visual Studio by selecting and installing the latest version of the package.

Next, you need to develop the required code. Fortunately, SendGrid provides clear and concise documentation, including a sample in C# that can be easily adapted for your purposes.

To implement the integration, you'd do the following:

1. Obtain an API key. (For convenience, it can be stored in an environment variable.)
2. Create a client object, which serves as the primary interface for interacting with the SendGrid API.
3. Define the sender and recipient addresses.
4. Construct a message object, populating it with the sender address, recipient address, subject, and body content.
5. Send the message using the client object.

By following these steps, you can seamlessly transition from the deprecated SMTP server to a modern SendGrid-based solution for email communication.

```
import SendGrid
import SendGrid.Helpers.Mail

subroutine sndeml
    sender, a
    senderEmail, a
    recipient, a
    recipientEmail, a
    subject, a
    message, a
    endparams
proc
    data apiKey = Environment.GetCommandEnvironmentVariable("SENDGRID_APIKEY")
    data client = new SendGridClient(apiKey)
    data fromAddress = new EmailAddress(senderEmail, sender)
    data toAddress = new EmailAddress(recipientEmail, recipient)
    data msg = MailHelper.CreateSingleEmail(fromAddress, toAddress, subject, message, mes
    client.SendEmailAsync(msg).Wait()
    xreturn
endsubroutine
```

Sample email implementation with SendGrid

Effortless integration with industry-standard APIs

As the example above shows, you can get a lot of mileage from a relatively small amount of effort. Talk about an efficient use of time as a developer! To implement the basics, you only needed to set up a SendGrid account, add a NuGet package, and write less than 10 lines of source code. Of course, that doesn't include tests and validation, but getting a functional API call in that short amount of time and code is exciting. Even better: you can run this same .NET code on Windows or Linux with Synergex's cross-platform .NET support.

SendGrid is just one example. It's very likely that any popular industry API you are interested in will have a pre-built .NET package for implementing it—including, Docusign, Paypal, and Microsoft Graph.

Synergy .NET gives your existing applications access to industry-standard API interfaces and opens the door to integrating in ways you might otherwise have avoided due to the amount of effort required. Since you can prototype a concept in just a few hours, you can confidently explore an idea without being afraid of potentially wasting months of development effort.

What would it look like to automate the signing of a report with DocuSign? How many manual hours would it save? What if you could integrate a payment provider like PayPal or Square? What benefit would that add for customers? What if you could sign into Microsoft Graph and have an alternate path to a user's files and activity? Or publish application trail history into Graph for use elsewhere in the products? It's easy to test the concepts.

Incremental adoption

.NET offers more ways for an application to deliver value, while only modifying the parts that will add value. Instead of attempting to convert all your software to .NET at once, you can take your core libraries and a couple of mainlines and build them in parallel in .NET. Only move software when it makes sense to do so.

Then grab a NuGet package (for SendGrid or SSH or .NET or PayPal), build a prototype, and iterate quickly on ideas for what your software can achieve. .NET opens the door to experimenting in new ways, regardless of whether you are operating on Windows or Linux.

You can now bring more value

- To your customers through new integrations with other service providers
- Inside your business by integrating with the tools, APIs, and software that drive your internal processes
- To your team by implementing cross-business unit requirements in data access, communication, and logging

And this can all happen with minimal conversion, quick iteration, and maximum value.

Moving traditional Synergy code to Synergy .NET

Synergex recognizes that migrating traditional Synergy code to Synergy .NET is a key strategic move for many customers, offering significant benefits (as outlined in this white paper), and we are fully committed to making this migration process as seamless as possible. Accordingly, we continually enhance the quality, performance, and compatibility of Synergy .NET, with a strong focus on keeping the tools easy to use and build with. Additionally, we closely track Microsoft's Visual Studio developments to ensure optimal integration and to deliver the latest capabilities and features as quickly as possible.

Strong typing

Something you'll likely encounter when moving to .NET is that .NET is more strict than traditional Synergy—which is a good thing. For example, .NET

- Enforces bounds checking
- Cleans up deprecated data types and syntax
- Prohibits circular references
- Enforces types on routine arguments

(For a complete list, refer to "Differences for Synergy .NET" in the Synergy/DE documentation.)

These stricter rules were put in place to ensure that source code would translate cleanly to a .NET strong type system. These enhancements will improve the quality of your codebase and help prevent potential runtime failures.

Approaching type enforcement incrementally is easy. The .NET compiler offers a few options to relax some parameter type restrictions, enabling you to get your software up and running in .NET quickly. These relaxed options are especially helpful with passing structures and passing alphas to decimals. So, even though .NET is strict, there are ways to adapt and improvements you can implement to facilitate its adoption.

Getting started with .NET

The best way to get started is to get the latest Visual Studio release, get the latest SDI release, and create some new projects. Say you have an existing traditional Synergy solution of multi-mainline projects, core libraries, and the like, and you want to migrate some of those programs to .NET. It's easy to take this kind of project structure to .NET, and both traditional and .NET code can keep working in the same solution. Here are the basic steps:

1. **Migrate your core library (all your supporting routines that your programs use).**
    a. Create a new project and select a Class Library (.NET) project type.
    b. Add links to the source files of your core library.
    c. If your program uses UI Toolkit, get and install Synergex's tklib NuGet package.
2. **Build your new core library .NET project.**
3. **Fix any .NET strong typing or other code issues.**
4. **Migrate at least one mainline.**
    a. Create a new project and select the Multiple Mainline (.NET) project type. (A multi-mainline is a project type dedicated to building up a group of programs that have common dependencies, so you don't need to have 100 different projects for 100 programs.)
    b. Add a link to the source file. (You'll want to make sure that if you make edits to the source, they'll be in both projects. That also means that any edits need to be compatible with traditional Synergy and .NET.)
    c. Make sure the .NET project gets all environment variables that are set up by the traditional projects. (Under Properties, Common Properties, turn on "Use common properties.")
5. **Build your mainline project and fix any strong typing or other code issues.**

You're all set! After setting up your code for .NET, the only thing limiting your application is your team's imagination.

For more information about moving to Synergy .NET, see "Synergy .NET Development" in our documentation.

CASE STUDY

Synergy .NET propels fuel industry app to business success

For over 40 years, Rural Computer Consultants, Inc. (RCC) has been a leader in software and services for the fuel and propane industry, offering a full range of software products that maximize efficiency for fuel delivery companies. In addition to their robust and versatile DBL-based Fuel Distribution System (FDS), which can be customized with over 20 modules, they offer mobile fuel management, cloud hosting, and computer hardware sales and support to fuel distributors.

Transformation in action

FDS was originally written in COBOL. In the late 1990s, RCC converted it to DBL to move to Windows, improve the look and feel, and address Y2K concerns. They updated their UI with Synergex's UI Toolkit, and their updated application served them well until competitors with shinier UIs started appearing in the late 2000s. That's when they began using Microsoft .NET and creating a new UI with WPF.

Synergex's release of Synergy .NET in 2010 was pivotal for RCC. They decided to move their back-end business logic to Synergy .NET, using an incremental approach that focused on the most used functions first.

This enabled them to create a variety of modern front-ends: a WPF client (C# and XAML), a WPF mobile work station for drivers and service techs on the road, and a web hub written in Angular.

With a .NET back end, they were able to develop comprehensive, modern front ends that could easily connect to their business logic. Besides opening opportunities for their application, the move to Synergy .NET provided a potentially larger benefit: it became easier to hire and onboard new developers. Says Nate Bahl, Head of Development at RCC, "With the move to Synergy .NET, we've been able to bring on new developers much quicker. If you know C#, you can pick up on Synergy .NET very quickly. With Visual Studio as their IDE, they're closing out tasks and fixing bugs by the end of their first days. It's been a big win for everybody."

With the help of Synergy .NET, RCC expanded its pioneering code base and logic to deliver a superior, modern customer experience and functionality. At the same time, they were able to strengthen their development team and position it for future growth.

Learn more about RCC's journey

Read the Case Study

Conclusion

.NET offers powerful opportunities to Synergy developers for expanding solutions and meeting business needs faster and more effectively. Migrating to Synergy .NET is a straightforward process that will strengthen your codebase. Whether building a new integration, enhancing a process, or tackling a larger initiative, by leveraging .NET, organizations can ensure their applications remain robust, versatile, and ready for the future.

Want to learn more?

For more information, contact Synergex at +1.916.635.7300 or synergy@synergex.com.

Get a boost from Synergex Professional Services Group

Our experts can help you evaluate .NET and get started on your journey.

Learn About PSG

Synergex provides software development tools, application integration technologies, and expert consulting services to help enterprise application developers retain their software investment, keep up with advancing technologies, and bring their applications into the future. For over 45 years, Synergex technologies have been the foundation of applications that drive commerce around the world. Every day, millions of users interact with these systems in e-commerce, global logistics, manufacturing, healthcare, and other industries.